[Primeira capa]

QUARTA EDIÇÃO

Redes de computadores

Andrew S. Tanenbaum

[Dísticos]

[1]Satélite de órbita baixa

[2]Camada de aplicação

[3]Comunicação sem fios

[4]LAN virtual

[5]Difusão sem fios

[6]Camada de transporte

[7]LAN sem fios

[8]Comutação da camada de enlace de dados

[9]Camada de rede

[10]Ponte

[11]Colisão

[12]Bem-vindo à Internet

[13]Túnel

[14]MAN sem fios

[15]Congestionamento

[16]Cliente

[17]Servidor

[18]GASOLINA

[19]Qualidade de Serviço

[20]Dificilmente algum serviço

[21]Serviços diferenciados

[22]Ethernet de gigabit

[23]Algoritmo de roteamento

[24]Pilha de protocolos

[25]E-mail

[26]Balde vazando

[27]A Web

[28]IP móvel

[29]Olá! Sou Harald

Editora Campus

===

[Quarta capa]

[Criação de redes de computadores]

[Dísticos]

[1] LAN sem fios

[2] MAN sem fios

[3] E-mail

QUARTA EDIÇÃO

Redes de computadores

Andrew S. Tanenbaum

A principal introdução mundial às redes — completamente atualizado de acordo com as tecnologias fundamentais de amanhã

Redes de computadores, quarta edição, é a introdução ideal às redes de hoje — e às redes de *amanhã*. Este best-seller clássico foi completamente atualizado para refletir as tecnologias mais novas e mais importantes de redes, com ênfase especial em redes sem fios, incluindo 802.11, Bluetooth™, comunicação sem fios de banda larga, redes *ad hoc*, i-mode e WAP. Porém as redes fixas não foram ignoradas, com cobertura de ADSL, Internet via cabo, Ethernet de gigabit, redes não hierárquicas, NAT e MPLS. Além disso, existe grande quantidade de material novo sobre aplicações, inclusive mais de 60 páginas sobre a Web, e ainda rádio na Internet, voz sobre IP e vídeo por demanda. Por fim, a cobertura de segurança de redes foi revista e expandida para preencher um capítulo inteiro.

Autor, educador e pesquisador, Andrew S. Tanenbaum, vencedor do ACM Karl V. Karlstrom Outstanding Educator Award, explica cuidadosamente como as redes funcionam *do lado de dentro*, desde o hardware subjacente na camada física até a camada de aplicação de nível superior. Tanenbaum focaliza todos estes temas e muitos outros:

[B] Camada física (cobre, fibra óptica, redes sem fios, satélites e Internet via cabo)

[B] Camada de link de dados (conceitos fundamentais de protocolos, verificação de protocolos, HDLC, SLIP e PPP)

[ej1] Comentário: substituir por link por enlace (camada de enlace de dados é um termo usual em português)
.

[B] Subcamada MAC (Ethernet de gigabit, 802.11, comunicação sem fios de banda larga e comutação)

[B] Camada de rede (algoritmos de roteamento, controle de congestionamento, QoS, IPv4 e IPv6)

[B] Camada de transporte (programação de soquetes, UDP, TCP, RTP e desempenho de redes)

[B] Camada de aplicação (e-mail, a Web, PHP, Web sem fios, MP3 e streaming audio)

[B] Segurança de redes (AES, RSA, criptografia quântica, IPsec e segurança da Web)

O livro fornece descrições detalhadas dos princípios associados a cada camada e apresenta muitos exemplos extraídos da Internet e de redes sem fios.

O Autor

ANDREW S. TANENBAUM é professor de ciência da computação na Vrije Universiteit em Amsterdam, Holanda, e diretor científico da ASCI, uma escola de pós-graduação holandesa fundada por universidades importantes de toda a Holanda. Ele também é membro do conselho do IEEE e membro do conselho da ACM. Outros livros de Tanenbaum, como autor ou co-autor, incluem *Structured Computer Organization*, quarta edição; *Operating Systems: Design and Implementation*, segunda edição; *Modern Operating Systems*, segunda edição; e *Distributed Systems: Principles and Paradigms* (Prentice Hall).

Visite a Editora Campus na Internet: http://www.campus.com.br
==

[TA]Prefácio

Este livro já está em sua quarta edição. Cada edição correspondeu a uma fase distinta na forma como as redes de computadores eram utilizadas. Quando a primeira edição americana foi lançada em 1980, as redes eram uma curiosidade acadêmica. Em 1988, ano da segunda edição, as redes estavam sendo usadas por universidades e grandes empresas. Na época em que a terceira edição foi lançada, em 1996, as redes de computadores, especialmente a Internet, já haviam se tornado uma realidade diária para milhões de pessoas. O novo item na quarta edição é o rápido crescimento das redes sem fio, em suas muitas formas.

O quadro das redes mudou radicalmente desde a terceira edição. Em meados da década de 1990, existiam numerosos tipos de LANs e WANs, além de várias pilhas de protocolos. Em 2003, a única LAN instalada com ampla utilização era a Ethernet, e virtualmente todas as LANs estavam na Internet. De acordo com isso, foi removida deste livro uma grande quantidade de material sobre essas redes mais antigas.

Entretanto, também há uma grande quantidade de novos desenvolvimentos. O mais importante é o enorme crescimento das redes sem fios, incluindo 802.11, loops locais sem fios, redes celulares 2G e 3G, Bluetooth, WAP, i-mode e outras. Acompanhando essa tendência, foi incluída neste volume uma grande quantidade de material sobre redes sem fios. Outro tópico que se tornou importante recentemente é a segurança; assim, foi acrescentado um capítulo inteiro sobre esse assunto.

Apesar de o Capítulo 1 ter a mesma função introdutória que tinha na terceira edição, o conteúdo foi completamente revisado e atualizado. Por exemplo, são dadas nesse capítulo introduções à Internet, Ethernet e LANs sem fios, juntamente com um pouco de história e fundamentos básicos. O capítulo também

discute brevemente as redes domésticas.

O Capítulo 2 foi um tanto reorganizado. Após uma breve introdução aos princípios de comunicação de dados, há três seções importantes sobre transmissão (meios guiados, sem fios e por satélite) seguidos por três outros em exemplos importantes (o sistema de telefonia pública comutada, o sistema de telefonia móvel e a televisão a cabo). Entre os novos tópicos abordados neste capítulo estão ADSL, difusão sem fios, MANs sem fios e acesso à Internet através de cabo e DOCSIS.

O Capítulo 3 sempre tratou dos princípios fundamentais de protocolos ponto a ponto. Essas idéias são essencialmente atemporais e não mudaram durante décadas. assim, a série de exemplos detalhados de protocolos apresentados neste capítulo permanece em grande parte inalterada desde a terceira edição.

Em contraste, a subcamada MAC tem sido uma área de grande atividade nos últimos anos; assim, muitas mudanças estão presentes no Capítulo 4. A seção sobre Ethernet foi expandida para incluir a Ethernet de gigabit. São completamente novas seções importantes sobre redes sem fios, difusão sem fios, Bluetooth e comutação da camada de enlace de dados, inclusive MPLS.

O Capítulo 5 também foi atualizado, com a remoção de todo o material sobre ATM e a inclusão de material adicional sobre a Internet. A qualidade do serviço também é agora um tópico importante, incluindo discussões de serviços integrados e serviços diferenciados. As redes sem fios também estão presentes aqui, como uma discussão do roteamento em redes *ad hoc*. Outros tópicos novos incluem NAT e redes não hierárquicas (peer-to-peer).

O Capítulo 6 ainda trata da camada de transporte, mas também há grandes mudanças. Entre elas encontra-se um exemplo de programação de soquetes. Um cliente de uma página e um servidor de uma página são apresentados em C e discutidos. Juntos, eles fornecem um servidor primitivo de arquivos remotos ou

[ej1] Comentário: o gigabit Ethernet

[ej2] Comentário: complet amente novas, são as seções importantes sobre

[ej3] Comentário: Um cliente e um servidor em código C de uma página são apresentados e discutidos.

da Web, disponível para experimentação. Outros novos tópicos incluem chamada de procedimentos remotos, RTP e transaction-TCP.

O Capítulo 7, que descreve a camada de aplicação, ficou mais nitidamente concentrado. Após uma curta introdução ao DNS, o restante do capítulo lida com apenas três tópicos: e-mail, a Web e multimídia. Porém, cada tópico é tratado com muitos detalhes. A descrição do funcionamento da Web tem agora mais de 60 páginas, cobrindo uma ampla variedade de tópicos, inclusive páginas da Web estáticas e dinâmicas, HTTP, scripts da CGI, redes de entrega de conteúdo, cookies e caches da Web. Também há material sobre como escrever páginas da Web modernas, incluindo breves introduções a XML, XSL, XTML, PHP e muito mais, todas com exemplos que podem ser testados. A Web sem fios também é discutida, focalizando o i-mode e o WAP. O material sobre multimídia inclui agora MP3, streaming audio, rádio pela Internet e voz sobre IP.

[ej4] Comentário: focado

A segurança se tornou tão importante que agora foi expandida para ocupar um capítulo completo com mais de 100 páginas. Ele abrange os princípios de segurança (algoritmos simétricos e de chave pública, assinaturas digitais e certificados X.509) e as aplicações desses princípios (autenticação, segurança de correio eletrônico e segurança na Web). O capítulo é ao mesmo tempo amplo (variando desde criptografia quântica até censura governamental) e profundo (por exemplo, com detalhes sobre o funcionamento do SHA-1).

O Capítulo 9 contém uma lista totalmente nova de leituras sugeridas e uma bibliografia completa, com mais de 350 citações sobre a literatura atual. Mais de 200 dessas citações se referem a artigos e livros escritos a partir do ano 2000.

Os livros de informática estão repletos de acrônimos. E este não é exceção. Quando tiver concluído a leitura deste volume, todas estas siglas terão um sentido claro para você: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML,

HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RPC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW e XML. Mas não se preocupe. Cada um desses acrônimos será cuidadosamente definido antes de ser usado.

Para ajudar os instrutores a utilizarem este livro como um texto para um curso de treinamento, o autor preparou os seguintes complementos para auxílio ao ensino, incluindo:

[B] Um manual de soluções de problemas.

[B] Arquivos contendo as figuras em vários formatos.

[B] Transparências do PowerPoint para um curso com a utilização do livro.

[B] Um simulador (escrito em C) para os exemplos de protocolos do Capítulo 3.

[B] Uma página da Web com links para muitos tutoriais, organizações, FAQs etc.

O manual de soluções está disponível diretamente na Prentice Hall (mas **somente** para instrutores, não para alunos). Todo o material restante encontra-se no Web site do livro:

http://www.prenhall.com/tanenbaum

Quando estiver lá, clique na capa do livro.

Muitas pessoas me ajudaram durante o curso da quarta edição. Gostaria de agradecer especialmente às seguintes pessoas: Ross Anderson, Elizabeth Belding-Royer, Steve Bellovin, Chatschik Bisdikian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Thomas Robertazzi,

Medy Sanadidi, Christian Schmutzer, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn e Maarten van Steen. Agradecimentos especiais a Trudy Levine, por provar que as avós são capazes de fazer um excelente trabalho de revisão de material técnico. Shivakant Mishra elaborou muitos problemas desafiantes para os finais de capítulos. Andy Dornan sugeriu leituras adicionais para o Capítulo 9. Jan Looyen forneceu hardware essencial em um momento crítico. O dr. F. de Nies fez um ótimo trabalho de recorte e colagem, exatamente quando foi necessário. Minha editora na Prentice Hall, Mary Franz, me ofereceu mais material de leitura do que eu havia consumido nos sete anos anteriores, e também foi útil em vários outros aspectos.

Por fim, chegamos às pessoas mais importantes: Suzanne, Barbara e Marvin. A Suzanne, por seu amor, sua paciência e seus almoços no campo. A Barbara e Marvin por serem divertidos e alegres o tempo todo (exceto quando reclamavam de alguns terríveis livros didáticos da faculdade, o que me manteve sempre atento). Obrigado.

Andrew S. Tanenbaum

[TA]O autor

Andrew S. Tanenbaum é bacharel em ciências pelo M.I.T e Ph.D. pela University of California em Berkeley. Atualmente, é professor de ciência da computação na Vrije Universiteit em Amsterdam, Holanda, onde lidera o Computer Systems Group. Ele também é decano da Advanced School for Computing and Imaging, um programa interuniversitário em nível de pós-graduação, que desenvolve pesquisas sobre sistemas paralelos avançados, distribuídos e de imagens. No entanto, Tanenbaum vem tentando de todas as formas não se tornar um burocrata.

No passado, ele desenvolveu pesquisas sobre compiladores, sistemas operacionais, interligação de redes e sistemas distribuídos locais. Sua área de pesquisa atual tem como foco principal o projeto e a implementação de sistemas geograficamente distribuídos que alcançam um bilhão de usuários. Essa pesquisa, que está sendo realizada em conjunto com o professor Maarten van Steen, é descrita em *www.cs.vu.nl/globe*. Em conjunto, todos esses projetos de pesquisa já produziram mais de 100 artigos em periódicos científicos e conferências, além de cinco livros.

O professor Tanenbaum também já produziu um considerável volume de software. Ele foi o principal projetista do Amsterdam Compiler Kit, um kit de ferramentas para o desenvolvimento de compiladores portáteis amplamente utilizado, como também do MINIX, um pequeno clone do UNIX, destinado ao uso em laboratórios de programação para estudantes. Esse sistema forneceu a inspiração e a base sobre a qual foi desenvolvido o Linux. Juntamente com seus alunos do curso de doutorado e outros programadores, ele ajudou a desenvolver o sistema operacional distribuído Amoeba, um sistema operacional distribuído baseado na utilização de um microkernel de alto desempenho. Agora, os sistemas

MINIX e Amoeba estão disponíveis gratuitamente na Internet.

Após concluírem o curso, geralmente seus alunos de doutorado seguem carreiras ainda mais brilhantes, o que o deixa muito orgulhoso.

O professor Tanenbaum é membro do conselho da ACM, membro do conselho do IEEE e membro da Royal Netherlands Academy of Arts and Sciences; em 1994, ele recebeu da ACM o Karl V. Karlstrom Outstanding Educator Award. Tanenbaum também recebeu em 1997 da ACM/SIGCSE o Award for Outstanding Contributions to Computer Science Education e o prêmio Texty de 2002 por excelência em livros didáticos. Ele também faz parte da lista *Who's Who in the World*. Sua home page na World Wide Web está no URL *http://www.cs.vu.nl/~ast/*.

Redes de computadores

Quarta edição

===

Outros títulos de sucesso de Andrew S. Tanenbaum

Distributed Systems: Principles and Paradigms

Esse novo livro, em co-autoria com Maarten van Steen, aborda tanto os conceitos fundamentais quanto os paradigmas dos modernos sistemas distribuídos. Na primeira parte, ele focaliza em detalhes os princípios de comunicação, processos, nomenclatura, sincronização, consistência e replicação, tolerância a falhas e segurança. Em seguida, na segunda parte, se aprofunda em diferentes paradigmas utilizados para elaborar sistemas distribuídos, inclusive sistemas orientados a objetos, sistemas de arquivos distribuídos, sistemas orientados a documentos e sistemas baseados em coordenação. Numerosos exemplos são discutidos extensivamente.

Modern Operating Systems, 2° edição

Esse texto completo estuda em detalhes os conceitos fundamentais dos modernos sistemas operacionais e os ilustra com numerosos exemplos reais. Após um capítulo introdutório, os cinco capítulos seguintes lidam com os conceitos básicos: processos e threads, impasses, gerenciamento de memória, entrada/saída e sistemas de arquivos. Os próximos seis capítulos lidam de material mais avançado, incluindo sistemas de multimídia, sistemas de vários processadores, segurança. Por fim, são apresentados dois estudos de casos detalhados: UNIX/Linux e Windows 2000.

[ej1] Comentário: Sistemas operacionais modernos

[ej2] Comentário: deadlocks

[ej3] Comentário: com

Structured Computer Organization, 4° edição

Esse clássico, um grande sucesso que está agora em sua quarta edição, oferece a introdução ideal à arquitetura de computadores. Ele aborda o assunto com uma estratégia de fácil compreensão, estudando-o de baixo para cima. Há um capítulo sobre lógica digital para iniciantes, seguido por capítulos sobre microarquitetura, sobre o nível de arquiteturas de conjuntos de instruções, sistemas operacionais, linguagem assembly e arquiteturas de computadores paralelos.

Operating Systems: Design and Implementation, 2° edição

Esse texto popular sobre sistemas operacionais, escrito em parceria com Albert S. Woodhull é o único livro que abrange os conceitos fundamentais de sistemas operacionais e também sua aplicação a um sistema real. Todos os tópicos tradicionais de sistemas operacionais são estudados em detalhes. Além disso, os conceitos básicos são cuidadosamente ilustrados com o MINIX, um sistema operacional gratuito baseado no POSIX, semelhante ao UNIX, para computadores pessoais. Cada exemplar inclui um CD-ROM gratuito, contendo o sistema MINIX completo, inclusive todo o código-fonte. O código-fonte está listado em um apêndice do livro e é explicado em detalhes no texto.

==

[Dedicatória]

A Suzanne, Barbara, Marvin e à memória de Bram e Sweetie π

==

Redes de computadores

Quarta edição

Andrew S. Tanenbaum

Vrije Universiteit

Amsterdam, Holanda

Tradução: Vandenberg D. de Souza

Analista de sistemas e tradutor

Editora Campus

==

[TA1]Capítulo

[TA2]1

[T1]Introdução

Cada um dos três séculos anteriores foi dominado por uma única tecnologia. O Século XVIII foi a época dos grandes sistemas mecânicos que acompanharam a Revolução Industrial. O Século XIX foi a era das máquinas a vapor. As principais conquistas tecnológicas do Século XX se deram no campo da aquisição, do processamento e da distribuição de informações. Entre outros desenvolvimentos, vimos a instalação das redes de telefonia em escala mundial, a invenção do rádio e da televisão, o nascimento e o crescimento sem precedentes da indústria de informática e o lançamento dos satélites de comunicação.

Como resultado do rápido progresso tecnológico, essas áreas estão convergindo rapidamente e são cada vez menores as diferenças entre coleta, transporte, armazenamento e processamento de informações. Organizações com centenas de escritórios dispersos por uma extensa área geográfica podem, com um simples apertar de um botão, examinar o status atual de suas filiais mais remotas. À medida que cresce nossa capacidade de colher, processar e distribuir informações, torna-se ainda maior a demanda por formas de processamento de informações ainda mais sofisticadas.

Apesar de a indústria de informática ainda ser jovem em comparação a outros setores industriais (por exemplo, o de automóveis e o de transportes aéreos), foi simplesmente espetacular o progresso que os computadores conheceram em um curto período de tempo. Durante as duas primeiras décadas de sua existência, os sistemas computacionais eram altamente centralizados, em geral instalados em uma grande sala com paredes de vidro, através das quais os visitantes podiam contemplar embevecidos aquela maravilha eletrônica. Uma empresa de médio

porte ou uma universidade contava apenas com um ou dois computadores, enquanto as grandes instituições tinham, no máximo, algumas dezenas. Era pura ficção científica a idéia de que, em apenas 20 anos, haveria milhões de computadores igualmente avançados do tamanho de um selo postal.

A fusão dos computadores e das comunicações teve uma profunda influência na forma como os sistemas computacionais eram organizados. O conceito de "centro de computação" como uma sala com um grande computador ao qual os usuários levam seu trabalho para processamento agora está completamente obsoleto. O velho modelo de um único computador atendendo a todas as necessidades computacionais da organização foi substituído pelas chamadas redes de computadores, nas quais os trabalhos são realizados por um grande número de computadores separados, mas interconectados. A estrutura e a organização dessas redes são os temas deste livro.

Ao longo do livro, utilizaremos a expressão "rede de computadores" quando quisermos mencionar um conjunto de computadores autônomos interconectados por uma única tecnologia. Dois computadores estão interconectados quando podem trocar informações. A conexão não precisa ser feita por um fio de cobre; também podem ser usadas fibras ópticas, microondas, ondas de infravermelho e satélites de comunicações. Existem redes em muitos tamanhos, modelos e formas, como veremos mais adiante. Embora possa parecer estranho para algumas pessoas, nem a Internet nem a World Wide Web é uma rede computadores. No final deste livro, deverá ficar claro o motivo dessa afirmação. A resposta simples é que a Internet não é uma única rede, mas uma rede de redes, e a Web é um sistema distribuído que funciona na Internet.

Existe na literatura uma considerável confusão entre uma rede de computadores e um **sistema distribuído.** A principal diferença entre eles é que, em um sistema distribuído, um conjunto de computadores independentes parece ser, para seus

usuários, um único sistema coerente. Em geral, ele tem um único modelo ou paradigma que apresenta aos usuários. Com freqüência, uma camada de software sobre o sistema operacional, chamada **middleware**, é responsável pela implementação desse modelo. Um exemplo bem conhecido de sistema distribuído é a **World Wide Web**, na qual tudo tem a aparência de um documento (uma página da Web).

Em uma rede de computadores, essa coerência, esse modelo e esse software estão ausentes. Os usuários ficam expostos às máquinas reais, sem qualquer tentativa por parte do sistema de fazer as máquinas parecerem e atuarem de modo coerente. Se as máquinas tiverem hardware diferente e sistemas operacionais distintos, isso será totalmente visível para os usuários. Se quiser executar um programa em uma máquina remota, o usuário terá de efetuar o logon nessa máquina e executar o programa lá.

Na prática, um sistema distribuído é um sistema de software instalado em uma rede. O software dá ao sistema um alto grau de coesão e transparência. Conseqüentemente, é o software (e em particular o sistema operacional) que determina a diferença entre uma rede e um sistema distribuído, não o hardware.

Apesar disso, há uma considerável sobreposição entre os dois assuntos. Por exemplo, os sistemas distribuídos e as redes de computadores precisam movimentar arquivos. A diferença está em quem é o responsável pela movimentação, o sistema ou o usuário.

Embora este livro seja basicamente dedicado a redes, muitos tópicos também são importantes em sistemas distribuídos. Para obter mais informações sobre sistemas distribuídos, consulte (Tanenbaum e Van Steen, 2002).

[T2] 1.1 Usos de redes de computadores

Antes de iniciarmos o exame detalhado das questões técnicas, vale a pena

dedicar algum tempo a explicar por que as pessoas estão interessadas em redes de computadores e com que finalidade essas redes podem ser usadas. Afinal, se ninguém estivesse interessado em redes de computadores, poucas delas seriam elaboradas. Começaremos com os usos tradicionais em empresas e para indivíduos, e depois passaremos aos desenvolvimentos mais recentes relacionadas a usuários móveis e a redes domésticas.

[T3] 1.1.1 Aplicações comerciais

Muitas empresas têm um número significativo de computadores. Por exemplo, uma empresa pode ter computadores separados para monitorar a produção, controlar os estoques e elaborar a folha de pagamento. Inicialmente, cada um desses computadores funcionava isolado dos outros mas, em um determinado momento, a gerência deve ter decidido conectá-los para poder extrair e correlacionar informações sobre a empresa inteira.

Em termos um pouco mais genéricos, a questão aqui é o **compartilhamento de recursos**, e o objetivo é tornar todos os programas, equipamentos e especialmente dados ao alcance de todas as pessoas na rede, independente da localização física do recurso e do usuário. Um exemplo óbvio e bastante disseminado é um grupo de funcionários de um escritório que compartilham uma impressora comum. Nenhum dos indivíduos realmente necessita de uma impressora privativa, e uma impressora de grande capacidade conectada em rede muitas vezes é mais econômica, mais rápida e de mais fácil manutenção que um grande conjunto de impressoras individuais.

Porém, talvez mais importante que compartilhar recursos físicos como impressoras, scanners e gravadores de CDs, seja compartilhar informações. Toda empresa de grande e médio porte e muitas empresas pequenas têm uma dependência vital de informações computadorizadas. A maioria das empresas tem

registros de clientes, estoques, contas a receber, extratos financeiros, informações sobre impostos e muitas outras informações on-line. Se todos os computadores de um banco sofressem uma pane, ele provavelmente não duraria mais de cinco minutos. Uma instalação industrial moderna, com uma linha de montagem controlada por computadores, não duraria nem isso. Hoje, até mesmo uma pequena agência de viagens ou uma firma jurídica com três pessoas depende intensamente de redes de computadores para permitir aos seus funcionários acessarem informações e documentos relevantes de forma instantânea.

No caso de empresas menores, todos os computadores provavelmente se encontram em um único escritório ou talvez em um único edifício; porém, no caso de empresas maiores, os computadores e funcionários podem estar dispersos por dezenas de escritórios e fábricas em muitos países. Apesar disso, um vendedor em Nova York às vezes poderia ter necessidade de acessar um banco de dados de estoque de produtos localizado em Cingapura. Em outras palavras, o mero fato de um usuário estar a 15.000 quilômetros de distância de seus dados não deve impedi-lo de usar esses dados como eles fossem dados locais. Resumindo, trata-se de uma tentativa de pôr fim à "tirania da geografia".

No mais simples dos termos, é possível imaginar que o sistema de informações de uma empresa consiste em um ou mais bancos de dados e em algum número de funcionários que precisam acessá-los remotamente. Nesse modelo, os dados são armazenados em poderosos computadores chamados **servidores**. Com freqüência, essas máquinas são instaladas e mantidas em um local central por um administrador de sistemas. Em contraste, os funcionários têm em suas escrivaninhas máquinas mais simples, chamadas **clientes**, com as quais eles acessam dados remotos, por exemplo, para incluir em planilhas eletrônicas que estão elaborando. (Algumas vezes, faremos referência ao usuário humano da

máquina cliente como o "cliente", mas deve ficar claro a partir do contexto se estamos nos referindo ao computador ou a seu usuário.) As máquinas clientes e servidores são conectadas entre si por uma rede, como ilustra a Figura 1.1. Observe que mostramos a rede como uma simples elipse, sem qualquer detalhe. Utilizaremos essa forma quando mencionarmos uma rede no sentido abstrato. Quando forem necessários mais detalhes, eles serão fornecidos.

[arte: ver original p. 4]

[Dísticos]

[1]Cliente

[2]Servidor

[3]Rede

[F]Figura 1.1

[FL] Uma rede com dois clientes e um servidor

Todo esse arranjo é chamado **modelo cliente/servidor.** Ele é amplamente usado e constitui a base da grande utilização da rede. Ele é aplicável quando o cliente e o servidor estão ambos no mesmo edifício (por exemplo, pertencem à mesma empresa), mas também quando estão muito distantes um do outro. Por exemplo, quando uma pessoa em sua casa acessa uma página na World Wide Web, é empregado o mesmo modelo, com o servidor da Web remoto fazendo o papel do servidor e o computador pessoal do usuário sendo o cliente. Sob a maioria das condições, um único servidor pode cuidar de um grande número de clientes.

Se examinarmos o modelo cliente/servidor em detalhes, veremos que há dois processos envolvidos, um na máquina cliente e um na máquina servidora. A comunicação toma a forma do processo cliente enviando uma mensagem pela rede ao processo servidor. Então, o processo cliente espera por uma mensagem em resposta. Quando o processo servidor recebe a solicitação, ele executa o

trabalho solicitado ou procura pelos dados solicitados e envia de volta uma resposta. Essas mensagens são mostradas na Figura 1.2.

[arte: ver original p. 5]

[Dísticos]

[1]Máquina cliente

[2]Processo cliente

[3]Solicitação

[4]Rede

[5]Resposta

[6]Máquina servidora

[7]Processo servidor

[F]Figura 1.2

[FL] O modelo cliente/servidor envolve solicitações e respostas

Um segundo objetivo da configuração de uma rede de computadores está relacionado às pessoas, e não às informações ou mesmo aos computadores. Uma rede de computadores pode oferecer um eficiente médio de comunicação entre os funcionários. Agora, virtualmente toda empresa que tem dois ou mais computadores tem o recurso de **correio eletrônico** (**e-mail**), que os funcionários utilizam de forma geral para suprir uma grande parte da comunicação diária. De fato, os funcionários trocam mensagens de e-mail sobre os assuntos mais corriqueiros, mas grande parte das mensagens com que as pessoas lidam diariamente não tem nenhum significado, porque os chefes descobriram que podem enviar a mesma mensagem (muitas vezes sem qualquer conteúdo) a todos os seus subordinados, bastando pressionar um botão.

[ej1] Comentário: meio de comunicação

Contudo, o e-mail não é a única forma de comunicação otimizada que as redes de computadores tornaram possível. Com uma rede, é fácil duas ou mais pessoas

que trabalham em locais muito distantes escreverem juntas um relatório. Quando um trabalhador faz uma mudança em um documento on-line, os outros podem ver a mudança imediatamente, em vez de esperarem vários dias por uma carta. Tal aceleração facilita a cooperação entre grupos de pessoas distantes entre si, o que antes era impossível.

Outra forma de comunicação auxiliada pelo computador é a videoconferência. Usando essa tecnologia, funcionários em locais distantes podem participar de uma reunião, vendo e ouvindo uns aos outros e até mesmo escrevendo em um quadro-negro virtual compartilhado. A videoconferência é uma ferramenta eficiente para eliminar o custo e o tempo anteriormente dedicado às viagens. Algumas vezes, dizemos que a comunicação e o transporte estão disputando uma corrida, e a tecnologia que vencer tornará a outra obsoleta.

Um terceiro objetivo para um número cada vez maior de empresas é realizar negócios eletronicamente com outras empresas, em especial fornecedores e clientes. Por exemplo, fabricantes de automóveis, aeronaves e computadores, entre outros, compram subsistemas de diversos fornecedores, e depois montam as peças. Utilizando redes de computadores, os fabricantes podem emitir pedidos eletronicamente, conforme necessário. A capacidade de emitir pedidos em tempo real (isto é, conforme a demanda) reduz a necessidade de grandes estoques e aumenta a eficiência.

Um quarto objeto que está começando a se tornar mais importante é o de realizar negócios com consumidores pela Internet. Empresas aéreas, livrarias e lojas de discos descobriram que muitos clientes apreciam a conveniência de fazer compras em casa. Conseqüentemente, muitos empresas fornecem catálogos de suas mercadorias e serviços on-line e emitem pedidos on-line. Espera-se que esse setor cresça rapidamente no futuro. Ele é chamado **comércio eletrônico** (**e-commerce**).

[ej2] Comentário: muitas empresas

[T3] 1.1.2 Aplicações domésticas

Em 1977, Ken Olsen era presidente da Digital Equipment Corporation, então o segundo maior fornecedor de computadores de todo o mundo (depois da IBM). Quando lhe perguntaram por que a Digital não estava seguindo a tendência do mercado de computadores pessoais, ele disse: "Não há nenhuma razão para qualquer indivíduo ter um computador em casa". A história mostrou o contrário, e a Digital não existe mais. Por que as pessoas compram computadores para usar em casa? No início, para processamento de textos e jogos; porém, nos últimos anos, esse quadro mudou radicalmente. Talvez agora a maior motivação seja o acesso à Internet. Alguns dos usos mais populares da Internet para usuários domésticos são:

1. Acesso a informações remotas.
2. Comunicação entre pessoas.
3. Entretenimento interativo.
4. Comércio eletrônico.

O acesso a informações remotas tem várias formas. Ele pode significar navegar na World Wide Web para obter informações ou apenas por diversão. As informações disponíveis incluem artes, negócios, culinária, governo, saúde, história, passatempos, recreação, ciência, esportes, viagens e muitos outros. A diversão surge sob tantas formas que não podemos mencionar, e também se apresenta em outras formas que é melhor não mencionarmos.

Muitos jornais são publicados on-line e podem ser personalizados. Por exemplo, às vezes é possível solicitar todas as informações sobre políticos corruptos, grandes incêndios, escândalos envolvendo celebridades e epidemias, mas dispensar qualquer notícia sobre esportes. Algumas vezes, é até mesmo possível transferir os artigos selecionados por download para o disco rígido enquanto

você dorme ou imprimi-los na sua impressora pouco antes do café da manhã. À medida que essa tendência continuar, ela causará desemprego maciço entre os jovens entregadores de jornais, mas as empresas jornalísticas gostam dela, porque a distribuição sempre foi o elo mais fraco na cadeia de produção inteira.

A próxima etapa além de jornais (e de revistas e periódicos científicos) é a biblioteca digital on-line. Muitas organizações profissionais, como ACM (*www.acm.org*) e IEEE Computer Society (*www.computer.org*), já têm muitos periódicos e anais de conferências on-line. Outros grupos estão seguindo com rapidez essa tendência. Dependendo do custo, tamanho e peso de notebooks com dimensões de livros, os livros impressos poderão se tornar obsoletos. Os céticos devem observar o efeito que a máquina de impressão teve sobre os manuscritos medievais com iluminuras.

Todas as aplicações anteriores envolvem interações entre uma pessoa e um banco de dados remoto repleto de informações. A segunda grande categoria de utilização de redes é a comunicação entre pessoas, basicamente a resposta do Século XXI ao telefone do Século XIX. O correio eletrônico (e-mail) já é usado diariamente por milhões de pessoas em todo o mundo e seu uso está crescendo rapidamente. Em geral, ele já contém áudio e vídeo, além de texto e imagens. O odor talvez demore um pouco mais.

Hoje em dia, qualquer adolescente é fanático pela @@@**troca de mensagens instantâneas**. Esse recurso, derivado do programa *talk* do UNIX, em uso desde de aproximadamente 1970, permite que duas pessoas digitem mensagens uma para a outra em tempo real. Uma versão dessa idéia para várias pessoas é a **sala de bate-papo** (ou **chat room**), em que um grupo de pessoas pode digitar mensagens que serão vistas por todos.

[ej3] Comentário: não é um termo técnico. Pode ser traduzido literalmente.

Newsgroups (grupos de notícias) mundiais, com discussões sobre todo tópico concebível, já são comuns entre grupos seletos de pessoas, e esse fenômeno

crescerá até incluir a população em geral. O tom dessas discussões, em que uma pessoa divulga uma mensagem e todos os outros participantes do newsgroup podem ler a mensagem, poderá variar de bem-humorado a inflamado. Diferentes das salas de bate-papo, os newsgroups não são de tempo real, e as mensagens são gravadas. Assim, por exemplo, quando alguém voltar das férias, todas as mensagens publicadas durante esse período estarão bem guardadas, esperando para serem lidas.

Outro tipo de comunicação entre pessoas recebe freqüentemente o nome de comunicação não hierárquica (peer-to-peer), com o objetivo de distingui-la do modelo cliente/servidor (Parameswaran *et al.*, 2001). Nessa forma de comunicação, indivíduos que constituem um grupo livre podem se comunicar com outros participantes do grupo, como mostra a Figura 1.3. Em princípio, toda pessoa pode se comunicar com uma ou mais pessoas; não existe nenhuma divisão fixa entre clientes e servidores.

[ej4] Comentário: Não-hierárquico está correto, mas o termo par-a-par também é usado e é melhor adaptado a sigla P2P.

[arte: ver original p. 7]

[F]Figura 1.3

[FL] Em um sistema não hierárquico não existem clientes e servidores fixos

A comunicação não hierárquica realmente alcançou o auge por volta de 2000 com um serviço chamado Napster que, em seu pico, teve mais de 50 milhões de fãs de música trocando todos os tipos de músicas, constituindo aquilo que provavelmente foi a maior violação de direitos autorais em toda a história registrada (Lam e Tan, 2001; Macedonia, 2000). A idéia era bastante simples. Os associados registravam em um banco de dados central mantido no servidor Napster a música que tinham em seus discos rígidos. Se queria uma canção, cada associado verificava no banco de dados quem tinha a canção e ia diretamente até o local indicado para obtê-la. Por não manter de fato nenhuma música em suas

máquinas, a Napster argumentou que não estava infringindo os direitos autorais de ninguém. Os tribunais não concordaram e fecharam o site e a empresa. Porém, a geração seguinte de sistemas não hierárquicos eliminou o banco de dados central, fazendo cada usuário manter seu próprio banco de dados local, além de fornecer uma lista de outras pessoas próximas associadas ao sistema. Um novo usuário pode então ir até qualquer associado para ver o que ele tem e obter uma lista de outros associados, com a finalidade de examinar outras músicas e outros nomes. Esse processo de pesquisa pode ser repetido indefinidamente, até constituir em um local um grande banco de dados do que existe fora desse local. Essa atividade seria tediosa para as pessoas, mas é especialmente adequada para computadores.

Também existem aplicações legais para comunicação não hierárquica. Por exemplo, aficionados que compartilham músicas de domínio público ou amostras de faixas liberadas por novos conjuntos musicais para fins de publicidade, famílias que compartilham fotografias, filmes e informações sobre a árvore genealógica, e adolescentes que participam de jogos on-line com várias pessoas. De fato, uma das aplicações mais populares de toda a Internet, o correio eletrônico, é inerentemente não hierárquica. Espera-se que essa forma de comunicação venha a crescer consideravelmente no futuro.

O crime eletrônico não se restringe a infrações de direitos autorais. Outra área agitada é a dos jogos de apostas eletrônicos. Os computadores têm simulado vários tipos de atividades durante décadas. Por que não simular máquinas caça-níqueis, jogos de roleta, mesas de vinte-e-um e outros equipamentos de jogos? Bem, porque isso é ilegal em muitos países. O grande problema é que o jogo é legal em muitos outros lugares (na Inglaterra, por exemplo) e os donos de cassinos perceberam o potencial para jogos de apostas pela Internet. Então, o que acontecerá se o jogador e o cassino estiverem em países diferentes, com leis

conflitantes? Essa é uma boa pergunta.

Outras aplicações orientadas a comunicações incluem a utilização da Internet para realizar chamadas telefônicas, além de videotelefonia e rádio pela Internet, três áreas de rápido crescimento. Outra aplicação é o ensino à distância (telelearning), que significa freqüentar aulas às 8 da manhã sem a inconveniência de ter de sair da cama. No final das contas, o uso de redes para aperfeiçoar a comunicação entre os seres humanos pode se mostrar mais importante que qualquer dos outros usos.

Nossa terceira categoria é o entretenimento, uma indústria enorme e que cresce mais e mais a cada dia. A aplicação fundamental nesse caso (aquela que deverá orientar todas as outras) é o vídeo por demanda. Dentro de aproximadamente uma década talvez seja possível selecionar qualquer filme ou programa de televisão, qualquer que seja a época ou país em que tenha sido produzido, e exibi-lo em sua tela no mesmo instante. Novos filmes poderão se tornar interativos e ocasionalmente o usuário poderá ser solicitado a interferir no roteiro (Macbeth deve matar Duncan ou aguardar o momento propício?), com cenários alternativos para todas as hipóteses. A televisão ao vivo também poderá se tornar interativa, com os telespectadores participando de programas de perguntas e respostas, escolhendo entre concorrentes e assim por diante.

Por outro lado, talvez a aplicação mais importante não seja o vídeo por demanda, mas sim os jogos. Já temos jogos de simulação em tempo real com vários participantes, como os de esconder em um labirinto virtual, e simuladores de vôo em que os jogadores de uma equipe tentam abater os jogadores da equipe adversária. Se os jogos forem praticados com óculos de proteção e imagens tridimensionais de qualidade fotográfica e movimentos em tempo real, teremos uma espécie de realidade virtual compartilhada em escala mundial.

Nossa quarta categoria é o comércio eletrônico no sentido mais amplo do termo.

A atividade de fazer compras em casa já é popular e permite ao usuário examinar catálogos on-line de milhares de empresas. Alguns desses catálogos logo oferecerão a possibilidade de obter um vídeo instantâneo sobre qualquer produto com um simples clique no nome do produto. Após a compra eletrônica de um produto, caso o cliente não consiga descobrir como usá-lo, poderá ser consultado o suporte técnico on-line.

Outra área em que o comércio eletrônico já é uma realidade é o acesso a instituições financeiras. Muitas pessoas já pagam suas contas, administram contas bancárias e manipulam seus investimentos eletronicamente. Sem dúvida, isso crescerá à medida que as redes se tornarem mais seguras.

Uma área que praticamente ninguém previu é a de brechós eletrônicos (e-brechó?). Leilões on-line de objetos usados se tornaram uma indústria próspera. Diferente do comércio eletrônico tradicional, que segue o modelo cliente/servidor, os leilões on-line se parecem mais com um sistema não hierárquico, uma espécie de sistema de consumidor para consumidor. Algumas dessas formas de comércio eletrônico utilizam pequenas abreviações baseadas no fato de que "to" e "2" têm a mesma pronúncia em inglês. As mais populares estão relacionadas na Figura 1.4.

[ej5] Comentário: O mercado livre eletrônico (e-mercado livre?)

[arte: ver original p. 9]

[T]Tabela

Abreviação	Nome completo	Exemplo
B2C	Business-to-consumer	Pedidos de livros on-line
B2B	Business-to-business	Fabricante de automóveis solicitando pneus a um fornecedor
G2C	Government-to-consumer	Governo distribuindo eletronicamente formulários de impostos
C2C	Consumer-to-consumer	Leilões on-line de produtos usados

P2P Peer-to-peer Compartilhamento de arquivos

[F]Figura 1.4

[FL] Algumas formas de comércio eletrônico

Sem dúvida a diversidade de usos de redes de computadores crescerá rapidamente no futuro, e é provável que esse crescimento se dê por caminhos que ninguém é capaz de prever agora. Afinal, quantas pessoas em 1990 previram que o fato de adolescentes digitarem tediosamente pequenas mensagens de texto em telefones celulares enquanto viajavam de ônibus seria uma imensa fábrica de dinheiro para as empresas de telefonia 10 anos depois? No entanto, o serviço de mensagens curtas é muito lucrativo.

[ej6] **Comentário:** (short message) é um termo muito usado, vale a pena manter o termo em inglês entre parênteses.

As redes de computadores podem se tornar imensamente importantes para pessoas que se encontram em regiões geográficas distantes, dando a elas o mesmo acesso a serviços que é oferecido às pessoas que vivem em uma grande cidade. O ensino à distância pode afetar de forma radical a educação; as universidades poderão ser nacionais ou internacionais. A telemedicina só agora está começando a se desenvolver (por exemplo, com o monitoramento remoto de pacientes), mas pode vir a ser muito mais importante. Porém, a aplicação fundamental poderá ser algo comum, como usar a @@@câmera da Web (webcam) no refrigerador para verificar se é preciso comprar leite no caminho do trabalho para casa.

[ej7] **Comentário:** câmera via web

[T3] 1.1.3 Usuários móveis

Computadores móveis, como notebooks e PDAs (personal digital assistants), constituem um dos segmentos de mais rápido crescimento da indústria de informática. Muitos usuários desses computadores têm máquinas de desktop no escritório e querem se manter conectados a essa base mesmo quando estão

longe de casa ou em trânsito. Tendo em vista que é impossível ter uma conexão por fios em automóveis e aviões, existe um grande interesse em redes sem fios. Nesta seção, examinaremos rapidamente alguns usos das redes sem fios.

As redes sem fios têm muitas utilidades. Um uso comum é o escritório portátil. Quando viajam, muitas vezes as pessoas querem usar seu equipamento eletrônico portátil para enviar e receber ligações telefônicas, fax e correio eletrônico, navegar pela Web, acessar arquivos remotos e se conectar a máquinas distantes. Além do mais, elas querem fazer isso enquanto se encontram em qualquer lugar do planeta. Por exemplo, nas conferências de informática de hoje, os organizadores muitas vezes configuram uma rede sem fio na área de conferência. Qualquer pessoa com um notebook e um modem sem fio pode simplesmente ligar o computador e se conectar à Internet, como se o computador estivesse ligado a uma rede de fiação. De modo semelhante, algumas universidades instalam redes sem fios no campus, para que os alunos possam se sentar debaixo das árvores e consultar o catálogo de fichas da biblioteca ou ler seu correio eletrônico.

As redes sem fios têm grande valor para frotas de caminhões, táxis, veículos de entrega e funcionários de serviços de assistência técnica, que precisam manter-se em contato com a base de operações da empresa. Por exemplo, em muitas cidades, os motoristas de táxi são homens de negócios independentes, em vez de serem funcionários de uma empresa de táxi. Em algumas dessas cidades, os táxis têm uma tela de vídeo que o motorista pode observar. Ao receber uma chamada, um despachante central digita os pontos de partida e destino. Essa informação é exibida nas telas de vídeo dos motoristas, e também é emitido um aviso sonoro. O primeiro motorista a pressionar um botão na tela de vídeo recebe a chamada.

As redes sem fios também são importantes para os militares. Se, de uma hora para outra, for necessário disputar uma guerra em qualquer lugar no mundo,

talvez não seja possível contar com a possibilidade de usar a infra-estrutura de rede local. Será melhor levar seu próprio equipamento de rede.

Embora as redes sem fios e a computação móvel freqüentemente tenham uma estreita relação, elas não são idênticas, como mostra a Figura 1.5. Aqui, observamos uma distinção entre redes **sem fios fixas** e **sem fios móveis.** Algumas vezes, até mesmo os computadores portáteis podem estar conectados por fios. Por exemplo, se um viajante conecta um notebook à tomada de telefone em um quarto de hotel, ele tem mobilidade sem precisar utilizar uma rede sem fio.

[arte: ver original p. 10]

[T]Tabela

Sem fios	**Móvel**	**Aplicações**
Não	Não	Computadores de desktop em escritórios
Não	Sim	Um notebook usado em um quarto de hotel
Sim	Não	Redes em edifícios mais antigos que não dispõem de fiação
Sim	Sim	Escritório portátil; PDA para registrar o estoque de uma loja

[F]Figura 1.5

[FL] Combinações de redes sem fios e computação móvel

Por outro lado, alguns computadores sem fio não são portáteis. Esse é o caso, por exemplo, das empresas sediadas em edifícios antigos, nos quais não há cabeamento de rede para conectar os computadores. Para instalar uma rede sem fio, essas empresas só precisarão adquirir uma pequena caixa com alguns componentes eletrônicos, retirá-la da embalagem e conectar o equipamento. Essa solução pode ser muito mais econômica do que instalar a fiação necessária no edifício.

Porém, é claro que também existem as verdadeiras aplicações sem fios móveis, que variam desde o escritório portátil até pessoas caminhando por uma loja com um PDA para fazer o levantamento do estoque. Nos aeroportos de grande movimento, os funcionários das locadoras de automóveis trabalham no estacionamento com computadores portáteis sem fios. Eles digitam o número da placa do automóvel que está sendo devolvido, e seus equipamentos portáteis, nos quais há uma impressora interna, entram em contato com o computador principal, acessam as informações sobre o aluguel e imprimem a conta na mesma hora.

À medida que a tecnologia sem fio se torna mais difundida, podem surgir numerosas outras aplicações. Vamos examinar rapidamente algumas das possibilidades. Parquímetros sem fios apresentam vantagens, tanto para usuários quanto para as administrações municipais. Esses equipamentos poderiam aceitar cartões de crédito ou débito, com verificação instantânea, pelo link sem fio. Quando o tempo de estacionamento expirasse, o equipamento poderia confirmar a presença de um automóvel (fazendo ecoar um sinal) e relatar o término do prazo à polícia. Estima-se que apenas as prefeituras das cidades dos Estados Unidos poderiam recolher 10 bilhões de dólares a mais dessa maneira (Harte *et al.*, 2000). Além disso, uma fiscalização melhor do estacionamento ajudaria a preservar o ambiente, pois os motoristas que soubessem que seu estacionamento ilegal certamente seria punido preferiram usar o transporte público.

Máquinas automáticas para venda de alimentos, bebidas e outros itens são encontradas em todo lugar. Porém, o alimento não entra nas máquinas por mágica. Periodicamente, alguém chega com um caminhão para reabastecê-las. Se as máquinas automáticas de venda emitissem um relatório sem fio uma vez por dia anunciando seu estoque atual, o motorista do caminhão saberia quais máquinas precisariam de reposição e que quantidade de produto seria necessária.

Essa informação poderia levar a um planejamento mais eficiente do roteiro. É claro que essas informações também poderiam ser transmitidas por uma linha telefônica padrão, mas equipar cada máquina automática de venda com uma conexão telefônica fixa para fazer uma única ligação por dia iria aumentar a despesa mensal fixa.

Outra área em que as redes sem fios poderiam economizar dinheiro é a da leitura de medidores de consumo de serviços de utilidade pública. Se os medidores de eletricidade, gás, água e outros existentes nos lares das pessoas informassem o consumo mensal por uma rede sem fios, não seria necessário contratar pessoas para fazer a leitura dos medidores. De modo semelhante, detectores de fumaça sem fios poderiam ligar para o corpo de bombeiros em vez de fazer um grande ruído (o que teria pouco valor se ninguém estivesse em casa). À medida que o custo dos dispositivos de rádio e do tempo no ar caírem, cada vez mais medições e relatórios serão realizados com redes sem fios.

[ej8] Comentário: da taxa de utilização (air time)

Uma área de aplicação bem diferente para redes sem fios é a fusão esperada de telefones celulares e PDAs, transformando-os em minúsculos computadores sem fios. Uma primeira tentativa nesse sentido foi a dos pequeninos PDAs sem fios que podiam exibir páginas da Web simples em suas telas ainda mais reduzidas. Esse sistema, chamado **WAP 1.0** (**Wireless Application Protocol — protocolo de aplicações sem fios**), foi malsucedido principalmente devido às telas microscópicas, à baixa largura de banda e ao serviço deficiente. Contudo, dispositivos e serviços mais novos funcionarão melhor com o WAP 2.0.

Uma área em que esses dispositivos podem se destacar é chamada m-commerce (mobile-commerce) (Senn, 2000). A força motriz por trás desse fenômeno consiste em um amálgama de fabricantes de PDAs sem fios e operadores de redes que estão tentando descobrir como obter uma fatia do comércio eletrônico. Uma de suas esperanças é usar PDAs sem fios em transações bancárias e compras.

Uma idéia é usar os PDAs sem fios como uma espécie de carteira eletrônica, autorizando pagamentos em lojas, em substituição ao dinheiro e aos cartões de crédito. O débito aparece então na conta do telefone celular. Do ponto de vista da loja, esse esquema pode poupar-lhes a maior parte das tarifas da empresa de cartões de crédito, o que pode significar uma porcentagem elevada. É claro que esse plano pode ter efeito contrário ao desejado, pois os clientes de uma loja poderiam usar seus PDAs para verificar os preços dos concorrentes antes de comprarem. Pior ainda, as empresas de telefonia poderiam oferecer PDAs com leitoras de códigos de barras que permitiriam a um cliente examinar um produto em uma loja e depois obter instantaneamente um relatório detalhado de onde mais ele poderia ser adquirido e a que preço.

Tendo em vista que o operador de rede sabe onde o usuário está, alguns serviços são intencionalmente dependentes da localização. Por exemplo, talvez seja possível procurar por uma livraria vizinha ou um restaurante chinês. Mapas portáteis são outra possibilidade, bem como previsões do tempo bastante específicas ("quando vai parar de chover em meu quintal?"). Sem dúvida surgirão muitas outras aplicações à medida que esses dispositivos se tornarem mais difundidos.

Uma enorme vantagem do m-comércio é que os usuários de telefones celulares se acostumaram a pagar por tudo (em contraste com os usuários da Internet, que esperam conseguir tudo de graça). Se um Web site da Internet cobrasse uma taxa para permitir a seus clientes efetuarem pagamentos com cartão de crédito, haveria uma imensa reclamação dos usuários. Se uma operadora de telefonia celular permitisse às pessoas pagarem por itens de uma loja usando o telefone e depois cobrassem uma tarifa por essa conveniência, provavelmente isso seria aceito como algo normal. O tempo dirá.

[ej9] Comentário: m-commerce, para manter a coerência com a introdução do termo.

Um pouco mais distantes no tempo encontram-se as redes pessoais e os

computadores que podem ser usados como objetos pessoais. A IBM desenvolveu um relógio que executa o Linux (incluindo o sistema de janelas X11) e que tem conectividade sem fio para a Internet, a fim de enviar e receber correio eletrônico (Narayanaswami *et al.*, 2002). No futuro, as pessoas poderão trocar cartões de visitas simplesmente mostrando seus relógios umas às outras. Computadores sem fio de uso pessoal podem dar às pessoas acesso a salas seguras, do mesmo modo que os cartões com tarjas magnéticas de hoje (talvez em combinação com um código PIN ou alguma medição biométrica). Esses relógios também poderão ser capazes de obter informações relevantes à localização atual do usuário (por exemplo, restaurantes locais). As possibilidades são infinitas.

Relógios inteligentes com rádios fazem parte de nosso espaço mental desde seu aparecimento nas tiras de quadrinhos de Dick Tracy em 1946. Porém, o que dizer da poeira inteligente? Os pesquisadores de Berkeley acondicionaram um computador sem fio em um cubo com 1 mm de aresta (Warneke *et al.*, 2001). As aplicações potenciais incluem controle de estoque, embalagens e até pequenos pássaros, roedores e insetos.

[ej10] Comentário: o acompanhamento de estoque, embalages e até de pequenos pássaros, roedores e insetos.

[T3] 1.1.4 Questões sociais

A ampla introdução das redes trouxe novos problemas sociais, éticos e políticos. Vamos apenas fazer uma rápida referência a alguns deles; seria preciso pelo menos um livro inteiro para fazer um estudo completo desses problemas. Uma característica popular de muitas redes são os newsgroups ou BBSs, a partir dos quais as pessoas podem trocar mensagens com indivíduos que têm os mesmos interesses. Quando são tratados apenas assuntos técnicos ou passatempos como jardinagem, não há muita polêmica.

Os problemas começam a vir à tona quando os newsgroups abordam temas mais palpitantes, como política, religião ou sexo. Os pontos de vista divulgados nesses

grupos podem ser altamente ofensivos para algumas pessoas. Pior ainda, elas podem não ser politicamente corretas. Além disso, as mensagens não estão obrigatoriamente limitadas ao texto. Fotografias coloridas de alta resolução e mesmo pequenos videoclipes já podem ser transmitidos com facilidade pelas redes de computadores. Algumas pessoas adotam a visão de que cada um sabe o que faz, mas outras acham que a publicação de certos tipos de materiais (por exemplo, ataques a determinados países ou religiões, pornografia etc.) é simplesmente inaceitável e tem de ser censurada. Diferentes países têm leis distintas e conflitantes sobre esse assunto. Assim, essa polêmica está ficando cada vez mais acirrada.

As pessoas abriram processos contra operadores de redes, partindo do princípio de que, a exemplo do que ocorre com os jornais e revistas, eles têm que assumir a responsabilidade pelo conteúdo do que publicam. A resposta inevitável é que uma rede é como uma companhia telefônica ou uma empresa de correios e não se pode esperar que ela censure seus usuários. No entanto, seria ainda mais grave o fato de que, temerosos com a possibilidade de serem processados, os operadores de rede começassem a excluir todas as mensagens que pudessem dar margem a algum tipo de processo judicial, cerceando dessa forma a liberdade de expressão dos usuários. Com certeza, essa discussão ainda irá perdurar por algum tempo.

Outra área polêmica envolve os direitos do empregado e do empregador. Muitas pessoas lêem e escrevem mensagens de correio eletrônico no ambiente de trabalho. Muitos empregadores afirmam que têm o direito de ler e até mesmo censurar as mensagens de seus funcionários, inclusive as que são enviadas a partir de um computador doméstico depois do expediente. Nem todos os empregados concordam com esse ponto de vista.

Ainda que os empregadores tenham poder sobre os funcionários, poderemos

dizer que esse relacionamento também se aplica às universidades em relação aos estudantes? E no caso das escolas secundárias em relação a seus alunos? Em 1994, a Carnegie-Mellon University decidiu vetar o acesso a diversos newsgroups relacionados a sexo porque, na opinião de sua diretoria, o material era inadequado para menores (ou seja, seus poucos alunos com menos de 18 anos). Essa polêmica ainda vai durar alguns anos.

Outro tópico importante é a relação entre o governo e os cidadãos. O FBI instalou um sistema em muitos provedores de serviços da Internet para bisbilhotar todas as mensagens de correio eletrônico de entrada e saída, em busca de fragmentos de interesse para a instituição (Blaze e Bellovin, 2000; Sobel, 2001; e Zacks, 2001). O sistema foi originalmente chamado **Carnivore**, mas a publicidade ruim fez com que ele fosse renomeado com a sigla aparentemente mais inocente DCS1000. No entanto, seu objetivo ainda é espionar milhões de pessoas, na esperança de encontrar informações sobre atividades ilegais. Infelizmente, a Quarta Emenda à Constituição dos Estados Unidos proíbe buscas do governo sem um mandado de busca. O fato dessas 54 palavras, escritas no Século XVIII, ainda terem algum peso no Século XXI é uma questão que poderá manter os tribunais ocupados até o Século XXII.

O governo não tem o monopólio das ameaças à privacidade das pessoas. O setor privado também faz sua parte. Por exemplo, pequenos arquivos chamados cookies que os navegadores da Web armazenam nos computadores dos usuários permitem que as empresas controlem as atividades desses usuários no ciberespaço e também podem permitir que números de cartões de crédito, números de CPF e outras informações confidenciais vazem pela Internet (Berghel, 2001).

As redes de computadores oferecem o potencial para o envio de mensagens anônimas. Em algumas situações, esse recurso pode ser desejável. Por exemplo,

ele proporciona um meio para alunos, soldados, trabalhadores e cidadãos denunciem o comportamento ilegal de professores, oficiais, superiores e políticos sem medo de possíveis represálias. Por outro lado, nos Estados Unidos e na maioria dos países democráticos, a lei permite especificamente às pessoas acusadas o direito de se confrontarem com o acusador perante o juiz. Acusações anônimas não podem ser usadas como evidências.

[ej11] Comentário: denunciarem ou para que ... denunciem

Em resumo, as redes de computadores, assim como a imprensa há cerca de 500 anos, permitem que os cidadãos comuns manifestem suas opiniões de um modo novo para platéias inteiramente diferentes. Essa nova liberdade traz em seu bojo uma série de questões sociais, políticas e morais.

Junto com o lado bom vem o lado ruim. A vida parece ser assim. A Internet torna possível encontrar informações com rapidez, mas uma grande parte dessas informações é incorreta, enganosa ou completamente equivocada. O aconselhamento médico que você conseguiu na Internet pode ter vindo de um ganhador do Prêmio Nobel ou de alguém que abandonou os estudos no ensino médio. As redes de computadores também introduziram novos tipos de comportamento anti-social e criminoso. O lixo de correio eletrônico (spam) se tornou parte de nossa vida porque as pessoas reúnem milhões de endereços de correio eletrônico e vendem esses endereços em CD-ROM para supostos negociantes. As mensagens de correio eletrônico incluindo conteúdo ativo (basicamente, programas ou macros que são executados na máquina do receptor) podem conter vírus capazes de causar devastação.

O roubo da identidade está se tornando um problema sério, pois os ladrões coletam informações suficientes sobre uma pessoa para obter cartões de crédito e outros documentos em nome da vítima. Finalmente, a capacidade de transmitir música e vídeo digital abriu a porta para violações maciças de direitos autorais, difíceis de capturar e punir.

Muitos desses problemas poderiam ser resolvidos se a indústria de informática levasse a sério a segurança dos computadores. Se todas as mensagens fossem criptografadas e autenticadas, seria mais difícil haver danos. Essa tecnologia está bem estabelecida e será estudada em detalhes no Capítulo 8. O problema é que os fornecedores de hardware e software sabem que a inclusão de recursos de segurança custa dinheiro, e seus clientes não buscam tais características. Além disso, um número substancial de problemas é causado por bugs de software, que ocorrem porque os fornecedores continuam a acrescentar mais e mais recursos a seus programas, o que inevitavelmente significa mais código e portanto mais bugs. Um imposto sobre novos recursos poderia ajudar, mas isso talvez dificultasse as vendas em poucos trimestres. Um programa de reembolso por software defeituoso talvez fosse ótimo, exceto pelo fato de levar à bancarrota toda a indústria de software no primeiro ano.

[T2] 1.2 Hardware de rede

Vamos desviar nossa atenção das aplicações e dos aspectos sociais das redes (a parte de diversão) para as questões técnicas relacionadas ao projeto de redes (a parte de trabalho). Não existe nenhuma taxonomia de aceitação geral na qual todas as redes de computadores possam ser classificadas, mas duas dimensões se destacam das demais: a tecnologia de transmissão e a escala. Vamos examinar cada uma delas.

Em termos gerais, há dois tipos de tecnologias de transmissão em uso disseminado nos dias de hoje:

1. Links de difusão.

2. Links ponto a ponto.

As **redes de difusão** têm apenas um canal de comunicação, compartilhado por todas as máquinas da rede. Mensagens curtas, que em determinados contextos

são chamadas **pacotes**, enviadas por qualquer máquina, são recebidas por todas as outras. Um campo de endereço dentro do pacote especifica o destinatário pretendido. Quando recebe um pacote, uma máquina verifica o campo de endereço. Se o pacote se destinar à máquina receptora, ela o processará; se for destinado a alguma outra máquina, o pacote será simplesmente ignorado.

Como uma analogia, imagine uma pessoa gritando no final do corredor que leva a uma série de salas: "Watson, venha cá. Preciso de você." Embora o pacote possa ser recebido (ouvido) por muitas pessoas, apenas Watson responderá. As outras pessoas irão ignorá-lo. Outra analogia é um anúncio em um aeroporto informando que todos os passageiros do vôo 644 devem se encaminhar ao portão 12 para embarque imediato.

Em geral, os sistemas de difusão também oferecem a possibilidade de endereçamento de um pacote a *todos* os destinos, com a utilização de um código especial no campo de endereço. Quando um pacote com esse código é transmitido, ele é recebido e processado por todas as máquinas da rede. Esse modo de operação é chamado **difusão** (**broadcasting**). Alguns sistemas de difusão também admitem a transmissão para um subconjunto das máquinas, o que se conhece como **multidifusão** (**multicasting**). Um esquema possível é reservar um bit para indicar a multidifusão. Os $n - 1$ bits de endereço restantes podem conter o número de um grupo. Cada máquina pode se "inscrever" em qualquer um ou em todos os grupos. Quando um pacote é enviado a um determinado grupo, ele é entregue a todas as máquinas inscritas nesse grupo.

Em contraste, as redes **ponto a ponto** consistem em muitas conexões entre pares de máquinas individuais. Para ir da origem ao destino, um pacote nesse tipo de rede talvez tenha de visitar primeiro uma ou mais máquinas intermediárias. Como normalmente é possível haver várias rotas com diferentes tamanhos, encontrar boas rotas é algo importante em redes ponto a ponto. Como regra geral (embora

existam muitas exceções), redes menores geograficamente localizadas tendem a usar difusão, enquanto redes maiores em geral são redes ponto a ponto. A transmissão ponto a ponto com um transmissor e um receptor às vezes é chamada unidifusão (unicasting).

[ej12] Comentário: o termo unidifusão é extremamente raro. Nesse caso, seria melhor deixar apenas unicasting.

Um critério alternativo para classificar as redes é sua escala. Na Figura 1.6, mostramos uma classificação de sistemas de vários processadores organizada por seu tamanho físico. Na parte superior encontram-se as **redes pessoais**, redes destinadas a uma única pessoa. Por exemplo, uma rede sem fios conectando um computador com o mouse, o teclado e a impressora é uma rede pessoal. Além disso, um PDA que controla o aparelho de audição ou o marcapasso de um usuário se enquadra nessa categoria. Além das redes pessoais, encontramos redes de maior abrangência. Essas redes podem ser divididas em redes locais, metropolitanas e geograficamente distribuídas (ou remotas). Finalmente, a conexão de duas ou mais redes é chamada inter-rede. A Internet mundial é um exemplo bastante conhecido de inter-rede. A distância é importante como uma métrica de classificação, porque são empregadas diferentes técnicas em escalas distintas. Neste livro, nos preocuparemos com as redes em todas essas escalas. Apresentaremos a seguir uma breve introdução ao hardware de rede.

[arte: ver original p. 16]

[Dísticos]

[1]Distância entre processadores

1 m

10 m

100 m

1 km

10 km

100 km

1.000 km

10.000 km

[2]Processadores localizados no(a) mesmo(a)

Metro quadrado

Sala

Edifício

Campus

Cidade

País

Continente

Planeta

[3]Exemplo

Rede pessoal

Rede local

Rede metropolitana

Rede geograficamente distribuída

A Internet

[F]Figura 1.6

[FL] Classificação de processadores interconectados por escala

[T3] 1.2.1 Redes locais

As **redes locais**, muitas vezes chamadas **LANs**, são redes privadas contidas em um único edifício ou campus universitário com até alguns quilômetros de extensão. Elas são amplamente usadas para conectar computadores pessoais e estações de trabalho em escritórios e instalações industriais de empresas, permitindo o compartilhamento de recursos (por exemplo, impressoras) e a troca de informações. As LANs têm três características que as distinguem de outros

tipos de redes: (1) tamanho, (2) tecnologia de transmissão e (3) topologia.

As LANs têm um tamanho restrito, o que significa que o pior tempo de transmissão é limitado e conhecido com antecedência. O conhecimento desse limite permite a utilização de determinados tipos de projetos que em outras circunstâncias não seriam possíveis, além de simplificar o gerenciamento da rede.

A tecnologia de transmissão das LANs quase sempre consiste em um cabo, ao qual todas as máquinas estão conectadas, como acontece com as linhas telefônicas compartilhadas que eram utilizadas em áreas rurais. As LANs tradicionais funcionam em velocidades de 10 Mbps a 100 Mbps, têm baixo retardo (microssegundos ou nanossegundos) e cometem pouquíssimos erros. As LANs mais modernas operam em até 10 Gbps. Neste livro, vamos aderir à tradição e medir as velocidades das linhas em megabits/s (1 Mbps correspondente a 1.000.000 bits/s) e gigabits/s (1 Gbps é igual a 1.000.000.000 bits/s).

As LANs de difusão admitem diversas topologias. A Figura 1.7 mostra duas delas. Em uma rede de barramento (isto é, um cabo linear), em qualquer instante no máximo uma máquina desempenha a função de mestre e pode realizar uma transmissão. Nesse momento, as outras máquinas serão impedidas de enviar qualquer tipo de mensagem. Então, será preciso criar um mecanismo de arbitragem para resolver conflitos quando duas ou mais máquinas quiserem fazer uma transmissão simultaneamente. O mecanismo de arbitragem pode ser centralizado ou distribuído. Por exemplo, o padrão IEEE 802.3, mais conhecido como **Ethernet**, é uma rede de difusão de barramento com controle descentralizado, em geral operando em velocidades de 10 Mbps a 10 Gbps. Os computadores em uma rede Ethernet podem transmitir sempre que desejam; se dois ou mais pacotes colidirem, cada computador aguardará um tempo aleatório e fará uma nova tentativa mais tarde.

[arte: ver original p. 17]

[Dísticos]

[1] Computador

[2] Cabo

(a)

[3] Computador

(b)

[F]Figura 1.7

[FL] Duas redes de difusão. (a) Barramento (b) Anel

Um segundo tipo de sistema de difusão é o anel. Em um anel, cada bit se propaga de modo independente, sem esperar pelo restante do pacote ao qual pertence. Em geral, cada bit percorre todo o anel no intervalo de tempo em que alguns bits são enviados, muitas vezes até mesmo antes de o pacote ter sido inteiramente transmitido. Assim como ocorre em todos os outros sistemas de difusão, existe a necessidade de se definir alguma regra para arbitrar os acessos simultâneos ao anel. São usados vários métodos, como fazer as máquinas adotarem turnos. O IEEE 802.5 (a rede Token Ring da IBM) é uma rede local baseada em anel que opera a 4 e 16 Mbps. O FDDI é outro exemplo de uma rede de anel.

As redes de difusão ainda podem ser divididas em estáticas e dinâmicas, dependendo do modo como o canal é alocado. Em uma alocação estática típica, o tempo seria dividido em intervalos discretos e seria utilizado um algoritmo de rodízio, fazendo com que cada máquina transmitisse apenas no intervalo de tempo de que dispõe. A alocação estática desperdiça a capacidade do canal quando uma máquina não tem nada a transmitir durante o intervalo de tempo (slot) alocado a ela, e assim a maioria dos sistemas procura alocar o canal dinamicamente (ou seja, à medida que é solicitado, ou por demanda).

Os métodos de alocação dinâmica de um canal comum são centralizados ou descentralizados. No método centralizado de alocação de canal, existe apenas uma entidade, por exemplo, uma unidade de arbitragem de barramento, que define quem transmitirá em seguida. Para executar essa tarefa, a entidade aceita solicitações e toma suas decisões de acordo com algum algoritmo interno. No método descentralizado de alocação de canal, não existe nenhuma entidade central; cada máquina deve decidir por si mesma se a transmissão deve ser realizada. Você poderia pensar que isso sempre leva ao caos, mas isso não acontece. Mais tarde, estudaremos muitos algoritmos criados para impedir a instauração do caos potencial.

[T3] 1.2.2 Redes metropolitanas

Uma **rede metropolitana**, ou **MAN**, abrange uma cidade. O exemplo mais conhecido de uma MAN é a rede de televisão a cabo disponível em muitas cidades. Esse sistema cresceu a partir de antigos sistemas de antenas comunitárias usadas em áreas com fraca recepção do sinal de televisão pelo ar. Nesses primeiros sistemas, uma grande antena era colocada no alto de colina próxima e o sinal era então conduzido até a casa dos assinantes.

Em princípio, esses sistemas eram sistemas *ad hoc* projetados no local. Posteriormente, as empresas começaram a entrar no negócio, obtendo concessões dos governos municipais para conectar por fios cidades inteiras. A etapa seguinte foi a programação de televisão e até mesmo canais inteiros criados apenas para transmissão por cabos. Com freqüência, esses canais eram altamente especializados, oferecendo apenas notícias, apenas esportes, apenas culinária, apenas jardinagem e assim por diante. Entretanto, desde sua concepção até o final da década de 1990, eles se destinam somente à recepção de televisão. A partir do momento em que a Internet atraiu uma audiência de massa, as

operadoras de redes de TV a cabo começaram a perceber que, com algumas mudanças no sistema, eles poderiam oferecer serviços da Internet de mão dupla em partes não utilizadas do espectro. Nesse momento, o sistema de TV a cabo começou a se transformar, passando de uma forma de distribuição de televisão para uma rede metropolitana. Em uma primeira aproximação, uma MAN seria semelhante ao sistema mostrado na Figura 1.8. Nessa figura, observamos que os sinais de televisão e da Internet são transmitidos ao @@@**head end** centralizado para distribuição subseqüente às casas das pessoas. Voltaremos a esse assunto, estudando-o em detalhes no Capítulo 2.

[ej13] Comentário: equipamento terminal central (head end). Esse termo geralmente não é traduzido.

[arte: ver original p. 18]

[Dísticos]

[1]Caixa de junção

[2]Antena

[3]Internet

[4]Head end

[F]Figura 1.8

[FL] Uma rede metropolitana baseada na TV a cabo

A televisão a cabo não é a única MAN. Os desenvolvimentos recentes para acesso à Internet de alta velocidade sem fio resultaram em outra MAN, que foi padronizada como IEEE 802.16. Estudaremos esse assunto no Capítulo 2.

[T3] 1.2.3 Redes geograficamente distribuídas

Uma **rede geograficamente distribuída**, ou **WAN** (wide area network), que também já foi denominada rede remota, abrange uma grande área geográfica, com freqüência um país ou continente. Ela contém um conjunto de máquinas cuja finalidade é executar os programas (ou seja, as aplicações) do usuário.

[ej14] Comentário: Isso não está no texto original, e a denominação é estranha.

Seguiremos a tradição e chamaremos essas máquinas de **hosts.** Os hosts estão conectados por uma **sub-rede de comunicação** ou, simplificando, uma **sub-rede.** Os hosts pertencem aos clientes (por exemplo, são os computadores pessoais dos usuários), enquanto a sub-rede de comunicação em geral pertence e é operada por uma empresa de telefonia ou por um provedor de serviços da Internet. A tarefa da sub-rede é transportar mensagens de um host para outro, exatamente como o sistema de telefonia transporta as palavras da pessoa que fala para a pessoa que ouve. Essa estrutura de rede é altamente simplificada, pois separa os aspectos da comunicação pura da rede (a sub-rede) dos aspectos de aplicação (os hosts).

[ej15] Comentário: usuários é melhor neste contexto.

[ej16] Comentário: computadores de uso pessoal.

Na maioria das redes geograficamente distribuídas, a sub-rede consiste em dois componentes distintos: linhas de transmissão e elementos de comutação. As **linhas de transmissão** transportam os bits entre as máquinas. Elas podem ser formadas por fios de cobre, fibra óptica, ou mesmo links de rádio. Os **elementos de comutação** são computadores especializados que conectam três ou mais linhas de transmissão. Quando os dados chegam a uma linha de entrada, o elemento de comutação deve escolher uma linha de saída para encaminhá-los. Esses computadores de comutação receberam diversos nomes no passado; o nome **roteador** é agora o mais comumente usado. Em inglês, algumas pessoas pronunciam esse nome da mesma forma que "rooter" e outras fazem rima com "doubter". A definição da pronúncia correta ficará como exercício para o leitor. (Observe que a resposta correta percebida talvez dependa da região em que reside o leitor.)

[ej17] Comentário: enlaces

Nesse modelo, mostrado na Figura 1.9, os hosts em geral estão conectados a uma LAN em que há um roteador, embora em alguns casos um host possa estar conectado diretamente a um roteador. O conjunto de linhas de comunicação e roteadores (sem os hosts) forma a sub-rede.

[arte: ver original p. 19]

[Dísticos]

[1]Sub-rede

[2]Roteador

[3]Host

[4]LAN

[F]Figura 1.9

[FL] Relação entre hosts em LANs e a sub-rede

Vale a pena fazermos um breve comentário em relação ao termo "sub-rede". Originalmente, seu **único** significado identificava o conjunto de roteadores e linhas de comunicação que transportava pacotes entre os hosts de origem e de destino. No entanto, alguns anos mais tarde, o termo adquiriu um segundo significado, em conjunto com o endereçamento de rede (que discutiremos no Capítulo 5). Infelizmente não existe nenhuma alternativa amplamente utilizada para seu significado inicial, e assim nós o utilizaremos com alguma hesitação em ambos os sentidos. Contudo, o contexto sempre deixará clara a acepção do termo que está sendo utilizado.

Na maioria das WANs, a rede contém numerosas linhas de transmissão, todas conectadas a um par de roteadores. No entanto, se dois roteadores que não compartilham uma linha de transmissão desejarem se comunicar, eles só poderão fazê-lo indiretamente, através de outros roteadores. Quando é enviado de um roteador para outro por meio de um ou mais roteadores intermediários, o pacote é recebido integralmente em cada roteador intermediário, onde é armazenado até a linha de saída solicitada ser liberada, para então ser encaminhado. Uma sub-rede organizada de acordo com esse principio é chamada sub-rede de **store-and-forward** (de armazenamento e encaminhamento) ou de **comutação por**

pacotes. Quase todas as redes geograficamente distribuídas (com exceção das que utilizam satélites) têm sub-redes store-and-forward. Quando são pequenos e têm todos o mesmo tamanho, os pacotes costumam ser chamados **células.**

O princípio de uma WAN de comutação por pacotes é tão importante que vale a pena dedicar mais algumas palavras a esse assunto. Em geral, quando um processo em algum host tem uma mensagem para ser enviada a um processo em algum outro host, primeiro o host que irá transmitir divide a mensagem em pacotes, cada um contendo seu número na seqüência. Esses pacotes são então injetados na rede um de cada vez em rápida sucessão. Os pacotes são transportados individualmente pela rede e depositados no host receptor, onde são novamente montados para formar a mensagem original, que é entregue ao processo receptor. Um fluxo de pacotes resultantes de alguma mensagem inicial é ilustrado na Figura 1.10.

[arte: ver original p. 20]

[Dísticos]

[1]Processo transmissor

[2]Host transmissor

[3]Roteador

[4]Sub-rede

[5]Host receptor

[6]Processo receptor

[7]O roteador C escolhe encaminhar pacotes para E e não para D

[8]Pacote

[F]Figura 1.10

[FL] Um fluxo de pacotes indo do transmissor até o receptor

Nessa figura, todos os pacotes seguem a rota *ACE*, em vez de *ABDE* ou *ACDE*. Em

algumas redes, todos os pacotes de uma determinada mensagem *devem* seguir a mesma rota; em outras, cada pacote é roteado separadamente. É claro que, se *ACE* for a melhor rota, todos os pacotes deverão ser enviados por ela, ainda que cada pacote seja roteado individualmente.

As decisões de roteamento são tomadas em caráter local. Quando um pacote chega ao roteador *A*, cabe ao roteador *A* decidir se esse pacote deve ser enviado na linha para *B* ou na linha para *C*. A forma como *A* toma essa decisão é chamada **algoritmo de roteamento**. Existem muitos desses algoritmos. Estudaremos alguns deles em detalhes no Capítulo 5.

Nem todas as WANs são comutadas por pacotes. Uma segunda possibilidade para uma WAN é um sistema de satélite. Cada roteador tem uma antena pela qual pode enviar e receber. Todos os roteadores podem ouvir a saída *do* satélite e, em alguns casos, eles também podem ouvir as transmissões de saída dos roteadores da mesma categoria *para* o satélite. Às vezes, os roteadores estão conectados a uma sub-rede ponto a ponto substancial, e apenas um deles tem uma antena de satélite. As redes de satélite são inerentemente redes de difusão e são mais úteis quando a propriedade de difusão é importante.

[ej18] **Comentário:** as transmissões

[ej19] **Comentário:** dos demais roteadores

[ej20] **Comentário:** de grande porte

[T3] 1.2.4 Redes sem fios

A comunicação digital sem fios não é uma idéia nova. Em 1901, o físico italiano Guglielmo Marconi demonstrou como funcionava um telégrafo sem fio que transmitia informações de um navio para o litoral por meio de código morse (afinal de contas, os pontos e traços são binários). Os modernos sistemas digitais sem fios têm um desempenho melhor, mas a idéia básica é a mesma.

Em uma primeira aproximação, redes sem fios podem ser divididas em três categorias principais:

1. Interconexão de sistemas.

2. LANs sem fios.

3. WANs sem fios.

A interconexão de sistemas significa interconectar os componentes de um computador usando rádio de alcance limitado. Quase todo computador tem um monitor, um teclado, um mouse e uma impressora, conectados por cabos à unidade principal. É tão grande o número de novos usuários que enfrentam grande dificuldade para conectar todos os cabos aos pequenos orifícios corretos (embora em geral eles sejam codificados com cores) que a maioria dos fabricantes de computadores oferece a opção de enviar um técnico à casa do usuário para fazê-lo. Conseqüentemente, algumas empresas se uniram para projetar uma rede sem fio de alcance limitado chamada **Bluetooth**, a fim de conectar esses componentes sem a utilização de fios. A rede Bluetooth também permite a conexão de câmeras digitais, fones de ouvido, scanners e outros dispositivos a um computador, simplesmente trazendo-os para dentro do alcance da rede. Nada de cabos, nada de instalação de drivers; basta juntá-los, ligá-los e eles funcionarão. Para muitas pessoas, essa facilidade de operação é uma grande vantagem.

Em sua forma mais simples, as redes de interconexão de sistemas utilizam o paradigma de mestre-escravo da Figura 1.11(a). A unidade do sistema é normalmente o mestre, comunicando-se com o mouse, o teclado etc., que atuam como escravos. O mestre informa aos escravos que endereços usar, quando eles podem transmitir, por quanto tempo podem transmitir, que freqüências podem usar e assim por diante. Discutiremos a rede Bluetooth com mais detalhes no Capítulo 4.

A próxima etapa em redes sem fios são as LANs sem fios. Elas são sistemas em que todo computador tem um modem de rádio e uma antena por meio dos quais pode se comunicar com outros sistemas. Freqüentemente, existe uma antena no

teto que permite a comunicação das máquinas, como mostra a Figura 1.11(b). Porém, se os sistemas estiverem próximos o bastante, eles poderão se comunicar diretamente um com o outro em uma configuração não hierárquica. As LANs sem fios estão se tornando cada vez mais comuns em pequenos escritórios e nos lares, onde a instalação da Ethernet é considerada trabalhosa demais, bem como em antigos edifícios comerciais, cantinas de empresas, salas de conferências e outros lugares. Existe um padrão para LANs sem fios, chamado **IEEE 802.11**, que a maioria dos sistemas implementa e que está se tornando bastante difundido. Discutiremos esse assunto no Capítulo 4.

[arte: ver original p. 22]

[Dísticos]

[1]Estação base

[2]Para a rede de fiação

[3] (a) (b)

[F]Figura 1.11

[FL] (a) Configuração da Bluetooth (b) LAN sem fios

O terceiro tipo de rede sem fio é usada em sistemas geograficamente distribuídos. A rede de rádio utilizada para telefonia celular é um exemplo de sistema sem fio de baixa largura de banda. Esse sistema já passou por três gerações. A primeira geração era analógica e usada apenas para voz. A segunda geração era digital e apenas para voz. A terceira geração é digital e se destina a voz e dados. Em certo sentido, as redes celulares sem fios são semelhantes às LANs sem fios, exceto pelo fato de que as distâncias envolvidas são muito maiores e as taxas de bits muito mais baixas. As LANs sem fios podem operar em velocidades de até 50 Mbps, sobre distâncias de dezenas de metros. Os sistemas celulares operam abaixo de 1 Mbps, mas a distância entre a estação base e o

computador ou telefone é medida em quilômetros, e não em metros. Teremos muito a dizer sobre essas redes no Capítulo 2.

Além dessas redes de baixa velocidade, também estão sendo desenvolvidas redes sem fios geograficamente distribuídas de alta largura de banda. O enfoque inicial é o acesso à Internet de alta velocidade sem fios a partir dos lares e de empresas comerciais, sem a utilização do sistema de telefonia. Esse serviço é chamado com freqüência serviço de distribuição local multiponto. Nós o estudaremos mais adiante no livro. Também foi desenvolvido um padrão para esse serviço, chamado IEEE 802.16. Examinaremos o padrão no Capítulo 4.

Quase todas as redes sem fios se conectam à rede de fiação em algum ponto, a fim de fornecer acesso a arquivos, bancos de dados e à Internet. Existem muitos modos de realizar essas conexões, dependendo das circunstâncias. Por exemplo, na Figura 1.12(a), representamos um avião com várias pessoas utilizando modems e telefones instalados nos assentos para se comunicar com o escritório. Cada chamada é independente das outras. Porém, uma opção muito mais eficiente é a LAN móvel da Figura 1.12(b). Aqui, cada assento é equipado com um conector Ethernet, ao qual os passageiros podem conectar seus computadores. Um único roteador na aeronave mantém um link de rádio com algum roteador no chão, trocando os roteadores à medida que o vôo prossegue. Essa configuração é simplesmente uma LAN tradicional, exceto pelo fato de que sua conexão para o mundo exterior é um link de rádio, em vez de uma linha de fiação.

[arte: ver original p. 23]

[Dísticos]

[1]Uma chamada telefônica por cada computador

[2]Computador portátil

[3]Roteador em vôo

[4]LAN fisicamente conectada

[5](a) (b)

[F]Figura 1.12

[FL] (a) Computadores móveis individuais (b) Uma LAN móvel

Muitas pessoas acreditam que as redes sem fios são a onda do futuro (por exemplo, Bi *et al.*, 2001; Leeper, 2001; Varshey e Vetter, 2000), mas existe pelo menos uma voz discordante. Bob Metcalfe, o inventor da Ethernet, escreveu: "Os computadores móveis sem fios são como banheiros móveis sem tubulação — verdadeiros penicos portáteis. Eles serão comuns em veículos, construções e em shows de rock. Meu conselho é que as pessoas instalem a fiação em suas casas e fiquem lá" (Metcalfe, 1995). A história deverá registrar esse comentário na mesma categoria da explicação dada em 1945 pelo presidente da IBM, T. J. Watson, ao discorrer sobre o motivo pelo qual a IBM não estava entrando no ramo de computadores: "Quatro ou cinco computadores devem ser suficientes para o mundo inteiro até o ano 2000".

[T3] 1.2.5 Redes domésticas

As redes domésticas estão surgindo no horizonte. A idéia fundamental é que, no futuro, a maioria dos lares estará configurada para redes. Todo dispositivo doméstico será capaz de se comunicar com cada um dos outros dispositivos, e todos eles estarão acessíveis pela Internet. Esse é um daqueles conceitos visionários que ninguém solicitou (como os controles remotos de TV ou os telefones celulares) mas, depois que chegaram, ninguém consegue mais imaginar como viver sem eles.

Muitos dispositivos são capazes de se conectar em rede. Algumas das categorias mais óbvias (com exemplos) são:

1. Computadores (PC de desktop, notebook, PDA, periféricos compartilhados).

[ej21] Comentário: mesa

2. Entretenimento (TV, DVD, videocassete, câmera de vídeo, câmera fotográfica, equipamento estéreo, MP3).

[ej22] Comentário: câmaras para monitorar bebês

3. Telecomunicações (telefone, celular, intercomunicador, fax).

4. Eletrodomésticos (microondas, refrigerador, relógio, forno, condicionador de ar, lâmpadas).

5. Telemetria (medidor de consumo de serviços de utilidade pública, alarme de fumaça/arrombamento, termostato, @@@babycam).

As redes de computadores domésticos já estão entre nós, em forma limitada. Muitos lares já têm um dispositivo para conectar vários computadores a uma conexão rápida da Internet. O entretenimento em rede ainda não é comum mas, à medida que uma quantidade cada vez maior de músicas e filmes puderem ser baixados da Internet, haverá demanda para conectar aparelhos estereofônicos e de televisão a ela. Além disso, as pessoas desejarão compartilhar seus próprios vídeos com amigos e com a família, e assim a conexão terá de ser bidirecional. O equipamento de telecomunicações já está conectado ao mundo exterior, mas logo ele será digital e acessará a Internet. O ambiente doméstico médio provavelmente terá uma dezena de relógios (por exemplo, nos eletrodomésticos), e todos terão de ser reajustados duas vezes por ano, ao se iniciar e ao terminar o período de horário de verão. Se todos os relógios estivessem conectados à Internet, esse reajuste poderia ser feito de forma automática. Por fim, o monitoramento remoto da casa e de seu conteúdo será a questão mais importante. Provavelmente muitos pais estariam dispostos a gastar algum dinheiro para monitorar o quarto de dormir do bebê em seus PDAs quando estiverem jantando fora, mesmo que uma babá esteja com a criança. Embora seja possível imaginar uma rede separada para cada área de aplicação, a integração de todas elas em uma única rede talvez seja uma idéia melhor.

As redes domésticas apresentam algumas propriedades fundamentalmente

distintas das características de outros tipos de redes. Primeiro, a rede e os dispositivos devem ser fáceis de instalar. O autor instalou numerosos itens de hardware e software em diversos computadores ao longo dos anos, com resultados variados. Uma série de chamadas telefônicas à assistência técnica do fornecedor em geral resultava em respostas como: (1) Leia o manual, (2) Reinicialize o computador, (3) Remova todo hardware e software exceto o nosso e tente outra vez, (4) Baixe o driver mais recente do nosso site da Web e, se tudo isso falhar, (5) Reformate o disco rígido e depois reinstale o Windows a partir do CD-ROM. Dizer ao comprador de um refrigerador com conexão para Internet que ele deve baixar e instalar uma nova versão do sistema operacional do refrigerador não deixar os clientes felizes. Os usuários de computadores estão acostumados a tolerar produtos que não funcionam; o público que adquire carros, televisores e refrigeradores é bem menos tolerante. Eles esperam produtos para funcionem 100% desde o início.

[ej23] Comentário: deixar á

[ej24] Comentário: que os produtos

Em segundo lugar, a rede e os dispositivos têm de ser à prova de falhas em sua operação. Os condicionadores de ar costumavam ter um único botão com quatro ajustes: DESLIGAR, BAIXO, MÉDIO e ALTO. Agora eles têm manuais de 30 páginas. Depois de ligados em rede, espera-se que apenas o capítulo sobre segurança ocupe 30 páginas. Isso estará além da compreensão de virtualmente todos os usuários.

Em terceiro lugar, o preço baixo é algo essencial para o sucesso. As pessoas não pagarão $50 a mais por um termostato capaz de se conectar à Internet, porque poucas pessoas consideram importante monitorar a temperatura de sua casa enquanto estão no trabalho. Porém, por $5 a mais, esse acessório teria boa aceitação.

Em quarto lugar, a principal aplicação provavelmente irá envolver recursos de multimídia, e assim a rede precisa ter capacidade suficiente. Não existe mercado

para televisores que mostram filmes trêmulos à resolução de 320 × 240 pixels e 10 quadros/s. A Fast Ethernet, o burro de carga na maioria dos escritórios, não tem qualidade suficiente para multimídia. Conseqüentemente, algumas redes precisarão de melhor desempenho do que as redes existentes nos escritórios, e a preços mais baixos, antes de se tornarem itens de mercado de massa.

Em quinto lugar, deve ser possível começar com um ou dois dispositivos e expandir gradualmente o alcance da rede. Isso significa que não deve haver guerras de formatos. Dizer aos consumidores para comprar periféricos com interfaces IEEE 1394 (FireWire) e alguns anos depois alegar que USB 2.0 é a interface do momento irá irritar os consumidores. A interface de rede terá de permanecer estável por muitos anos; a conexão física (se houver) terá de continuar estável durante décadas.

Em sexto lugar, segurança e confiabilidade serão fatores muito importantes. Perder alguns arquivos por causa de um vírus de correio eletrônico é uma coisa; ver um assaltante usar um PDA para desarmar seu sistema de segurança e depois saquear sua casa é algo bastante diferente.

Uma questão interessante é saber se as redes domésticas estarão fisicamente conectadas ou serão redes sem fios. A maioria das casas já tem seis redes instaladas: as de eletricidade, telefonia, TV a cabo, água, gás e esgoto. Acrescentar uma sétima rede durante a construção não é difícil, mas remodelar casas existentes é muito dispendioso. O custo favorece às redes sem fios, mas a segurança favorece às redes fisicamente conectadas. O problema das redes sem fios é que as ondas de rádio que elas utilizam passam com facilidade pelas cercas. Nem todo mundo fica satisfeito com a idéia de ter os vizinhos pegando carona em sua conexão da Internet e lendo suas mensagens de correio eletrônico a caminho da impressora. No Capítulo 8, estudaremos como a criptografia pode ser utilizada para proporcionar segurança mas, no contexto de uma rede

doméstica, a segurança tem de ser infalível, mesmo com usuários experientes.

[ej25] Comentário: a prova de tolos

[ej26] Comentário: inexperientes

Isso é algo mais fácil de dizer do que fazer, até mesmo no caso de usuários altamente sofisticados.

Em suma, as redes domésticas oferecem muitas oportunidades e muitos desafios. A maior parte desses desafios se relaciona à necessidade de ser fácil de administrar, confiável e segura, especialmente nas mãos de usuários não técnicos, ao mesmo tempo que oferece alto desempenho a baixo custo.

[T3] 1.2.6 Inter-redes

Existem muitas redes no mundo, com freqüência apresentando diferentes tipos de hardware e software. Normalmente, as pessoas conectadas a redes distintas precisam se comunicar entre si. Para que esse desejo se torne uma realidade, é preciso que se estabeleçam conexões entre redes quase sempre incompatíveis, às vezes por meio de máquinas chamadas **gateways**, que estabelecem a conexão e fazem a conversão necessária, tanto em termos de hardware quanto de software. Um conjunto de redes interconectadas é chamado **inter-rede** ou **internet**. Esses termos serão usados em um sentido genérico, em contraste com a Internet mundial (uma inter-rede específica), que sempre será representada com inicial maiúscula.

Uma forma comum de inter-rede é um conjunto de LANs conectadas por uma WAN. Na verdade, se resolvêssemos substituir o termo "sub-rede" da Figura 1.9 por "WAN", essa seria a única mudança que precisaríamos fazer na figura. Nesse caso, a única distinção técnica real entre uma sub-rede e uma WAN seria a presença (ou a ausência) de hosts. Se o sistema dentro da área em cor cinza contiver apenas roteadores, ele será uma sub-rede; se contiver roteadores e hosts, será uma WAN. As diferenças reais estão relacionadas à propriedade e ao uso.

Em geral, sub-redes, redes e inter-redes se confundem. Uma sub-rede faz mais sentido no contexto de uma rede geograficamente distribuída, onde ela se refere ao conjunto de roteadores e linhas de comunicação pertencentes à operadora da rede. Como analogia, o sistema telefônico consiste em estações de comutação telefônica conectadas entre si por linhas de alta velocidade e às casas e aos escritórios por linhas de baixa velocidade. Essas linhas e equipamentos, cuja propriedade e gerenciamento são da empresa de telefonia, formam a sub-rede do sistema telefônico. Os telefones propriamente ditos (os hosts nessa analogia) não fazem parte da sub-rede. A combinação de uma sub-rede e seus hosts forma uma rede. No caso de uma LAN, os cabos e os hosts formam a rede. Na verdade, não existe uma sub-rede.

Uma inter-rede é formada quando diferentes redes estão interconectadas. No nosso ponto de vista, a conexão de uma LAN e uma WAN ou a conexão de duas LANs forma uma inter-rede, mas ainda não existe um consenso na indústria quanto à terminologia a ser usada nessa área. Uma regra prática é que, se diferentes organizações pagam pela construção de partes distintas da rede e cada uma mantém sua parte, temos uma inter-rede, e não uma única rede. Além disso, se a tecnologia subjacente é diferente em partes distintas (por exemplo, difusão *versus* ponto a ponto), provavelmente temos duas redes.

[T2] 1.3 Software de rede

No projeto das primeiras redes de computadores, o hardware foi a principal preocupação e o software ficou em segundo plano. Essa estratégia foi deixada para trás. Atualmente, o software de rede é altamente estruturado. Nas próximas seções, examinaremos com algum detalhe a técnica de estruturação do software. O método descrito aqui é de fundamental importância para o livro inteiro e

faremos repetidas referências a ele.

[T3] 1.3.1 Hierarquias de protocolos

Para reduzir a complexidade do projeto, a maioria das redes é organizada como uma pilha de **camadas** ou **níveis**, colocadas umas sobre as outras. O número de camadas, o nome, o conteúdo e a função de cada camada diferem de uma rede para outra. No entanto, em todas as redes o objetivo de cada camada é oferecer determinados serviços às camadas superiores, isolando essas camadas dos detalhes de implementação desses recursos. Em certo sentido, cada camada é uma espécie de máquina virtual, oferecendo determinados serviços à camada situada acima dela.

Na realidade, esse conceito é familiar e é utilizado em toda a ciência da computação, na qual é conhecido por nomes diferentes, como ocultação de informações, tipos de dados abstratos, encapsulamento de dados e programação orientada a objetos. A idéia fundamental é que um determinado item de software (ou hardware) fornece um serviço a seus usuários, mas mantém ocultos os detalhes de seu estado interno e de seus algoritmos.

A camada *n* de uma máquina se comunica com a camada *n* de outra máquina. Coletivamente, as regras e convenções usadas nesse diálogo são conhecidas como o protocolo da camada *n*. Basicamente, um **protocolo** é um acordo entre as partes que se comunicam, estabelecendo como se dará a comunicação. Como uma analogia, quando uma mulher é apresentada a um homem, ela pode estender a mão para ele que, por sua vez, pode apertá-la ou beijá-la, dependendo, por exemplo, do fato de ela ser uma advogada americana que esteja participando de uma reunião de negócios ou uma princesa européia presente a um baile de gala. A violação do protocolo dificultará a comunicação, se não a tornar completamente impossível.

A Figura 1.13 ilustra uma rede de cinco camadas. As entidades que ocupam as camadas correspondentes em diferentes máquinas são chamadas **pares** (**peers**). Os pares podem ser processos, dispositivos de hardware ou mesmo seres humanos. Em outras palavras, são os pares que se comunicam utilizando o protocolo.

[arte: ver original p. 27]

[Dísticos]

[1] Host 1 Host 2

Camada 5 Protocolo da camada 5 Camada 5

[2]Interface entre as camadas 4 e 5

[3] Camada 4 Protocolo da camada 4 Camada 4

[4] Interface entre as camadas 3 e 4

[5] Camada 3 Protocolo da camada 3 Camada 3

[6] Interface entre as camadas 2 e 3

[7] Camada 2 Protocolo da camada 2 Camada 2

[8] Interface entre as camadas 1 e 2

[9] Camada 1 Protocolo da camada 1 Camada 1

[10] Meio físico

[F]Figura 1.13

[FL] Camadas, protocolos e interfaces

Na realidade, os dados não são transferidos diretamente da camada *n* de uma máquina para a camada *n* de outra máquina. Em vez disso, cada camada transfere os dados e as informações de controle para a camada imediatamente abaixo dela, até ser alcançada a camada mais baixa. Abaixo da camada 1 encontra-se o **meio físico** através do qual se dá a comunicação propriamente dita. Na Figura 1.13, a comunicação virtual é mostrada por linhas pontilhadas e a comunicação física por

linhas contínuas.

Entre cada par de camadas adjacentes existe uma **interface.** A interface define as operações e os serviços que a camada inferior tem a oferecer à camada que se encontra acima dela. Quando os projetistas de rede decidem a quantidade de camadas que será incluída em uma rede e o que cada uma delas deve fazer, uma das considerações mais importantes é a definição de interfaces claras entre as camadas. Por sua vez, isso exige que cada camada execute um conjunto específico de funções bem definidas. Além de reduzir o volume de informações que deve ser passado de uma camada para outra, as interfaces bem definidas simplificam a substituição da implementação de uma camada por uma implementação completamente diferente (por exemplo, a substituição de todas as linhas telefônicas por canais de satélite), pois a nova implementação só precisa oferecer exatamente o mesmo conjunto de serviços à sua vizinha de cima, assim como era feito na implementação anterior. De fato, é comum hosts diferentes utilizarem implementações distintas.

Um conjunto de camadas e protocolos é chamado **arquitetura de rede.** A especificação de uma arquitetura deve conter informações suficientes para permitir que um implementador desenvolva o programa ou construa o hardware de cada camada, de forma que ela obedeça corretamente ao protocolo adequado. Nem os detalhes da implementação nem a especificação das interfaces pertencem à arquitetura, pois tudo fica oculto dentro das máquinas e não é visível do exterior. Não é nem mesmo necessário que as interfaces de todas as máquinas de uma rede sejam iguais, desde que cada uma delas possa usar todos os protocolos de maneira correta. Uma lista de protocolos usados por um determinado sistema, um protocolo por camada, é chamada **pilha de protocolos.** Os principais assuntos deste livro serão as arquiteturas de rede, as pilhas de protocolos e os protocolos propriamente ditos.

Uma analogia pode ajudar a explicar a idéia de uma comunicação em vários níveis. Imagine dois filósofos (processos pares da camada 3), um dos quais fala urdu e inglês e o outro fala chinês e francês. Como não falam um idioma comum, eles contratam tradutores (processos pares da camada 2), que por sua vez têm cada qual uma secretária (processos pares da camada 1). O filósofo 1 deseja transmitir sua predileção por *oryctolagus cuniculus* a seu par. Para tal, ele envia uma mensagem (em inglês) através da interface 2/3 a seu tradutor, na qual diz "I like rabbits", como mostra a Figura 1.14. Como os tradutores resolveram usar um idioma neutro, o holandês, a mensagem foi convertida para "Ik vind konijnen leuk". A escolha do idioma é o protocolo da camada 2, que deve ser processada pelos pares da camada 2.

O tradutor entrega a mensagem a uma secretária para ser transmitida, por exemplo, pelo fax (o protocolo da camada 1). Quando chega, a mensagem é traduzida para o francês e passada através da interface 2/3 para o filósofo 2. Observe que cada protocolo é totalmente independente dos demais, desde que as interfaces não sejam alteradas. Nada impede que os tradutores mudem do holandês para o finlandês, desde que ambos concordem com a modificação e que ela não afete sua interface com a camada 1 ou com a camada 3. De modo semelhante, as secretárias também podem passar de fax para correio eletrônico ou telefone sem incomodar (ou mesmo informar) as outras camadas. Cada processo só pode adicionar informações dirigidas a seu par. Essas informações não são enviadas à camada superior.

Vejamos agora um exemplo mais técnico: como oferecer comunicação à camada superior da rede de cinco camadas da Figura 1.15. Uma mensagem *M* é produzida por um processo de aplicação que funciona na camada 5 e é entregue à camada 4 para transmissão. A camada 4 coloca um **cabeçalho** no início da mensagem para identificá-la e envia o resultado à camada 3. O cabeçalho inclui informações de

controle, como números de seqüência, a fim de permitir à camada 4 da máquina de destino repassar as mensagens na ordem correta, caso as camadas inferiores não mantenham a seqüência. Em algumas camadas, os cabeçalhos contêm ainda tamanho, hora e outros campos de controle.

[arte: ver original p. 29]

[Dísticos]

[1]Local A

3

2

1

[2]I like rabbits

[3]I: holandês

Ik vind konijnen leuk

[4]Fax #---

I: holandês

Ik vind konijnen leuk

[5]Mensagem

[6]Informações para o tradutor remoto

[7]Informações para a secretária remota

[8]Filósofo

[9]Tradutor

[10]Secretária

[11]Local B

3

2

1

[12]J'aime bien les lapins

[13]I: holandês

Ik vind konijnen leuk

[14]Fax #---

I: holandês

Ik vind konijnen leuk

[F]Figura 1.14

[FL] A arquitetura filósofo-tradutor-secretária

Em muitas redes, não há limite para o tamanho das mensagens transmitidas no protocolo da camada 4, mas quase sempre há um limite imposto pelo protocolo da camada 3. Conseqüentemente, a camada 3 deve dividir as mensagens recebidas em unidades menores, pacotes, anexando um cabeçalho da camada 3 a cada pacote. Nesse exemplo, M é dividido em duas partes, M_1 e M_2.

A camada 3 define as linhas de saída que serão usadas e transmite os pacotes à camada 2. A camada 2 acrescenta não apenas um cabeçalho à cada fragmento, mas também um final, e fornece a unidade resultante à camada 1 para transmissão física. Na máquina receptora, a mensagem se move de baixo para cima, de camada em camada, com os cabeçalhos sendo retirados durante o processo. Nenhum dos cabeçalhos das camadas abaixo de n é repassado à camada n.

[arte: ver original p. 30]

[Dísticos]

[1]Camada

[2]Protocolo da camada 5

[3]Protocolo da camada 4

[4]Protocolo da camada 3

[5]Protocolo da camada 2

[6]Máquina de origem

[7]Máquina de destino

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 1.15

[FL] Exemplo de fluxo de informações que admite a comunicação virtual na camada 5

Para entender a Figura 1.15 é importante observar a relação entre a comunicação virtual e a comunicação real, e a diferença entre protocolos e interfaces. Por exemplo, para os processos pares da camada 4, sua comunicação é "horizontal", utilizando o protocolo da camada 4. O procedimento de cada um deles tem um nome semelhante a *EnviarParaOutroLado* e *ReceberDoOutroLado*, muito embora esses procedimentos na realidade se comuniquem com camadas inferiores através da interface 3/4, e não com o outro lado.

A abstração de processos pares (peers) é fundamental para toda a estrutura da rede. Com sua utilização, a tarefa não gerenciável de projetar a rede completa pode ser dividida em diversos problemas de projeto menores e gerenciáveis, ou seja, o projeto das camadas individuais.

Embora o título da Seção 1.3 seja "Software de rede", vale a pena lembrar que as camadas inferiores de uma hierarquia de protocolos costumam ser implementadas no hardware ou no firmware. No entanto, algoritmos de protocolo muito complexos estão envolvidos no processo, muito embora estejam embutidos (parcial ou totalmente) no hardware.

[T3]1.3.2 Questões de projeto relacionadas às camadas

Algumas questões fundamentais de projeto que ocorrem em redes de computadores estão presentes em diversas camadas. Mencionaremos a seguir algumas das questões mais importantes.

Todas as camadas precisam de um mecanismo para identificar os transmissores e os receptores. Como em geral uma rede tem muitos computadores, e alguns deles têm vários processos, é necessário um meio para que um processo de uma máquina especifique com quem ela deseja se comunicar. Como existem vários destinos, surge a necessidade de se criar uma forma de **endereçamento** para definir um destino específico.

Outra preocupação que se deve ter em relação ao conjunto de decisões de projeto diz respeito à transferência de dados. Em alguns sistemas, os dados são transferidos em apenas um sentido; em outros, os dados trafegam em ambos os sentidos. O protocolo também deve definir a quantos canais lógicos corresponde a conexão e quais são suas prioridades. Muitas redes fornecem pelo menos dois canais lógicos por conexão, um para dados normais e um para dados urgentes.

O **controle de erros** é uma questão importante, pois os circuitos de comunicação física não são perfeitos. Muitos códigos de detecção e correção de erros são conhecidos, mas as partes envolvidas na conexão devem chegar a um consenso quanto ao que está sendo usado. Além disso, o receptor deve ter algum meio para informar ao transmissor quais mensagens foram recebidas corretamente e quais não foram.

Nem todos os canais de comunicação preservam a ordem das mensagens enviadas a eles. Para lidar com uma possível perda de seqüência, o protocolo deve permitir explicitamente ao receptor remontar de forma adequada os fragmentos recebidos. Uma solução óbvia é numerar os fragmentos, mas essa solução ainda deixa aberta a questão do que deve ser feito com os fragmentos que chegarem fora de ordem.

Uma questão que afeta cada nível é como impedir que um transmissor rápido envie uma quantidade excessiva de dados a um receptor mais lento. Várias soluções foram propostas e serão discutidas a seguir. Algumas delas envolvem uma espécie de feedback do receptor para o transmissor, seja direta ou indiretamente, sobre a situação atual do receptor. Outras limitam o transmissor a uma velocidade de transmissão predeterminada. Esse assunto é chamado **controle de fluxo.**

Outro problema a ser resolvido em diversos níveis é a falta de habilidade de todos os processos para aceitar mensagens arbitrariamente longas. Essa propriedade nos leva ao uso de mecanismos para desmontar, transmitir e remontar mensagens. Uma questão relacionada é o que fazer quando os processos insistem em transmitir dados em unidades tão pequenas que o envio de cada uma separadamente se torna ineficiente. Nesse caso, a solução é reunir as pequenas mensagens com um destino comum em uma grande mensagem e desmembrá-la na outra extremidade.

Quando for inconveniente ou dispendioso configurar uma conexão isolada para cada par de processos de comunicação, a camada subjacente pode decidir usar a mesma conexão para diversas conversações não relacionadas entre si. Desde que essa **multiplexação** e **demultiplexação** seja feita de forma transparente, ela poderá ser utilizada por qualquer camada. Por exemplo, a multiplexação é necessária na camada física, onde todo tráfego correspondente a todas as conexões tem de ser transmitido através de no máximo alguns circuitos físicos.

Quando houver vários caminhos entre a origem e o destino, uma rota deverá ser escolhida. Algumas vezes, essa decisão deve ser compartilhada por duas ou mais camadas. Por exemplo, para transmitir dados de Londres para Roma deve ser tomada uma decisão de alto nível (o trajeto passando pela França ou pela Alemanha, de acordo com suas respectivas leis de privacidade). Em seguida, é

preciso tomar uma decisão de baixo nível, a fim de selecionar um dos circuitos disponíveis de acordo com a carga de tráfego atual. Esse tópico é chamado **roteamento**.

[T3]1.3.3 Serviços orientados a conexões e serviços sem conexões

As camadas podem oferecer dois tipos diferentes de serviços às camadas situadas acima delas: serviços orientados a conexões e serviços sem conexões. Nesta seção, examinaremos esses dois tipos de serviços e as diferenças que existem entre eles.

O **serviço orientado a conexões** se baseia no sistema telefônico. Para falar com alguém, você tira o fone do gancho, disca o número, fala e, em seguida, desliga. Da mesma forma, para utilizar um serviço de rede orientado a conexões, primeiro o usuário do serviço estabelece uma conexão, utiliza a conexão, e depois libera a conexão. O aspecto essencial de uma conexão é que ela funciona como um tubo: o transmissor empurra objetos (bits) em uma extremidade, e esses objetos são recebidos pelo receptor na outra extremidade. Na maioria dos casos, a ordem é preservada, de forma que os bits chegam na seqüência em que foram enviados. Em alguns casos, quando uma conexão é estabelecida, o transmissor, o receptor e a sub-rede conduzem uma **negociação** sobre os parâmetros a serem usados, como o tamanho máximo das mensagens, a qualidade do serviço exigida e outros questões. Em geral, um lado faz uma proposta e a outra parte pode aceitá-la, rejeitá-la ou fazer uma contraproposta.

Por outro lado, o **serviço sem conexão** se baseia no sistema postal. Cada mensagem (carta) carrega o endereço de destino completo e cada uma delas é roteada (encaminhada) através do sistema, independentemente de todas as outras. Em geral, quando duas mensagens são enviadas ao mesmo destino, a primeira a ser enviada é a primeira a chegar. No entanto, é possível que a

primeira mensagem a ser enviada seja retardada, de modo que a segunda mensagem chegue primeiro.

Cada serviço pode ser caracterizado por uma **qualidade de serviço.** Alguns serviços são confiáveis, no sentido de nunca perderem dados. Em geral, um serviço confiável é implementado para que o receptor confirme o recebimento de cada mensagem, de modo que o transmissor se certifique de que ela chegou. O processo de confirmação introduz overhead e retardos, que freqüentemente compensam, mas às vezes são indesejáveis.

Uma situação típica em que um serviço orientado a conexões confiável é apropriado é a transferência de arquivos. O proprietário do arquivo deseja se certificar de que todos os bits chegaram corretamente e na mesma ordem em que foram enviados. São poucos os clientes de transferência de arquivos que preferem um serviço que ocasionalmente desorganiza ou perde alguns bits, mesmo que ele seja muito mais rápido.

O serviço orientado a conexões confiável tem duas pequenas variações secundárias: seqüências de mensagens e fluxos de bytes. Na primeira variação, os limites das mensagens são preservados. Quando duas mensagens de 1024 bytes são enviadas, elas chegam como duas mensagens distintas de 1024 bytes, nunca como uma única mensagem de 2048 bytes. Se as páginas de um livro forem enviadas por uma rede a uma fotocompositora como mensagens separadas, talvez seja importante preservar os limites da mensagem. Por outro lado, quando um usuário se conecta a um servidor remoto, só é necessário um fluxo de bytes do computador do usuário para o servidor. Os limites de mensagens não são relevantes.

Como já dissemos, para algumas aplicações, os retardos introduzidos pelas confirmações são inaceitáveis. Uma dessas aplicações é o tráfego de voz digital. Os usuários de telefone preferem ouvir um pouco de ruído na linha ou uma

palavra truncada de vez em quando a experimentar um retardo para aguardar confirmações. O mesmo acontece durante a transmissão de uma conferência de vídeo; não haverá problema se aparecerem alguns pixels errados. No entanto, é irritante ter de interromper o fluxo de transmissão para corrigir erros.

Nem todas as aplicações precisam de conexões. Por exemplo, à medida que o correio eletrônico se tornar mais comum, o lixo eletrônico também se tornará mais comum. Provavelmente, o transmissor de lixo de correio eletrônico não desejará enfrentar o problema de configurar e depois desfazer uma conexão apenas para enviar um item. Além disso, a não será essencial uma entrega 100% confiável, em especial se o custo for maior. É necessário apenas um modo de enviar uma única mensagem que tenha uma alta probabilidade de chegar, mas nenhuma garantia. O serviço sem conexão não confiável (ou seja, sem confirmação) costuma ser chamado **serviço de datagramas**, em uma analogia com o serviço de telegramas, que também não oferece uma confirmação ao transmissor.

Em outras situações, a conveniência de não ter de estabelecer uma conexão para enviar uma única mensagem curta é desejável, mas a confiabilidade é essencial. O **serviço de datagramas** com confirmação pode ser oferecido para essas aplicações. Ele é semelhante a enviar uma carta registrada e solicitar um aviso de recebimento. Quando o aviso é devolvido, o transmissor fica absolutamente certo de que a carta foi entregue ao destinatário e não foi perdida ao longo do caminho.

Outro serviço é o **serviço de solicitação/resposta.** Nele, o transmissor envia um único datagrama contendo uma solicitação; a resposta contém a réplica. Por exemplo, nessa categoria se enquadra uma consulta à biblioteca local perguntando onde se fala o idioma uighur. A solicitação/resposta em geral é usada para implementar a comunicação no modelo cliente/servidor: o cliente

emite uma solicitação e o servidor responde. A Figura 1.16 resume os tipos de serviços descritos anteriormente.

[arte: ver original p. 33]

[T]Tabela

	Serviço	**Exemplo**
Orientados a conexões		
	Fluxo de mensagens confiável	Seqüência de páginas
Fluxo de bytes confiável	Logon remoto	
	Conexão não confiável	Voz digitalizada
Sem conexões	Datagrama não confiável	Lixo de correio eletrônico
	Datagrama confirmado	Correspondência registrada
	Solicitação/resposta	Consulta a banco de dados

[F]Figura 1.16

[FL]Seis diferentes tipos de serviços

O conceito de usar comunicação não confiável pode ser confuso a princípio. Afinal de contas, por que alguém iria preferir uma comunicação não confiável à comunicação confiável? Em primeiro lugar, a comunicação confiável (em nosso sentido, isto é, confirmada) pode não estar disponível. Por exemplo, a Ethernet não fornece comunicação confiável. Ocasionalmente, os pacotes podem ser danificados em trânsito. Cabe aos níveis mais altos do protocolo lidar com esse problema. Em segundo lugar, os retardos inerentes ao fornecimento de um serviço confiável podem ser inaceitáveis, em especial nas aplicações de tempo real como as de multimídia. Por essas razões, coexistem tanto a comunicação confiável quanto a não confiável.

[T3]1.3.4 Primitivas de serviço

Um serviço é especificado formalmente por um conjunto de **primitivas** (operações) disponíveis para que um processo do usuário acesse o serviço. Essas primitivas informam ao serviço que ele deve executar alguma ação ou relatar uma ação executada por uma entidade par. Se a pilha de protocolos estiver localizada no sistema operacional, como ocorre com freqüência, as primitivas serão normalmente chamadas do sistema. Essas chamadas geram uma armadilha para o modo de núcleo que então devolve o controle da máquina ao sistema operacional para enviar os pacotes necessários.

O conjunto de primitivas disponíveis depende da natureza do serviço que está sendo fornecido. As primitivas para um serviço orientado a conexões são diferentes das que são oferecidas em um serviço sem conexões. Como um exemplo mínimo das primitivas de serviço que poderiam ser fornecidas para implementar um fluxo de bytes confiável em um sistema cliente/servidor, considere as primitivas listadas na Figura 1.17.

[arte: ver original p. 34]

[T]Tabela

Primitiva	**Significado**
LISTEN	Bloco que espera por uma conexão de entrada
CONNECT	Estabelecer uma conexão com um par que está à espera
RECEIVE	Bloco que espera por uma mensagem de entrada
SEND	Enviar uma mensagem ao par
DISCONNECT	Encerrar uma conexão

[F]Figura 1.17

[FL]Cinco primitivas de serviço para implementação de uma conexão simples

Essas primitivas poderiam ser usadas como a seguir. Primeiro, o servidor executa LISTEN para indicar que está preparado para aceitar conexões de entrada. Um

caminho comum para implementar LISTEN é torná-la uma chamada de sistema de bloqueio do sistema. Depois de executar a primitiva, o processo servidor fica bloqueado até surgir uma solicitação de conexão.

Em seguida, o processo cliente executa CONNECT para estabelecer uma conexão com o servidor. A chamada de CONNECT precisa especificar a quem se conectar; assim, ela poderia ter um parâmetro fornecendo o endereço do servidor. Em geral, o sistema operacional envia então um pacote ao par solicitando que ele se conecte, como mostra o item (1) na Figura 1.18. O processo cliente é suspenso até haver uma resposta. Quando o pacote chega ao servidor, ele é processado pelo sistema operacional do servidor. Quando o sistema observa que o pacote está solicitando uma conexão, ele verifica se existe um ouvinte. Nesse caso, ele realiza duas ações: desbloqueia o ouvinte e envia de volta uma confirmação (2). A chegada dessa confirmação libera o cliente. Nesse momento, o cliente e o servidor estão em execução e têm uma conexão estabelecida entre eles. É importante notar que a confirmação (2) é gerada pelo próprio código do protocolo, e não em resposta a uma primitiva no nível do usuário. Se chegar uma solicitação de conexão e não houver nenhum ouvinte, o resultado será indefinido. Em alguns sistemas, o pacote pode ser enfileirado por curto período antes de uma LISTEN.

A analogia óbvia entre esse protocolo e a vida real ocorre quando um consumidor (cliente) liga para o gerente do serviço de atendimento ao consumidor de uma empresa. O gerente de serviço inicia a seqüência ficando próximo ao telefone para atendê-lo, caso ele toque. Então, o cliente efetua a chamada. Quando o gerente levanta o fone do gancho, a conexão é estabelecida.

[arte: ver original p. 35]

[Dísticos]

[1]Processo cliente

Sistema operacional

[2]Máquina cliente

Chamadas do sistema

Núcleo Pilha de protocolos Drivers

[3](1) Solicitação de conexão

(2) ACK

(3) Solicitação de dados

(4) Resposta

(5) Desconexão

(6) Desconexão

[4]Máquina servidora

Processo servidor

Núcleo Pilha de protocolos Drivers

[F]Figura 1.18

[FL] Pacotes enviados em uma interação cliente/servidor simples, em uma rede orientada a conexões

A próxima etapa é a execução de RECEIVE pelo servidor, a fim de se preparar para aceitar a primeira solicitação. Normalmente, o servidor faz isso imediatamente após ser liberado de LISTEN, antes da confirmação poder retornar ao cliente. A chamada de RECEIVE bloqueia o servidor.

Depois, o cliente executa SEND para transmitir sua solicitação (3), seguida pela execução de RECEIVE para receber a resposta.

A chegada do pacote de solicitação à máquina servidora desbloqueia o processo servidor, para que ele possa processar a solicitação. Depois de terminar o trabalho, ele utiliza SEND para enviar a resposta ao cliente (4). A chegada desse pacote desbloqueia o cliente, que pode agora examinar a resposta. Se tiver

solicitações adicionais, o cliente poderá fazê-las nesse momento. Ao terminar, ele utilizará DISCONNECT para encerrar a conexão. Em geral, uma DISCONNECT inicial é uma chamada de bloqueio, suspendendo o cliente e enviando um pacote ao servidor para informar que a conexão não é mais necessária (5). Quando o servidor recebe o pacote, ele próprio também emite uma DISCONNECT, confirmando o pacote do cliente e liberando a conexão. Quando o pacote do servidor (6) volta à máquina cliente, o processo cliente é liberado e a conexão é interrompida. Em resumo, é assim que funciona a comunicação orientada a conexões.

É claro que a vida não é tão simples assim. Muitos detalhes podem sair errados. O sincronismo pode estar incorreto (por exemplo, CONNECT ser executada antes de LISTEN), os pacotes podem ser perdidos e muito mais. Examinaremos todas essas questões com muitos detalhes mais adiante; porém, por enquanto, a Figura 1.18 resume o funcionamento possível de uma comunicação cliente/servidor em uma rede orientada a conexões.

Considerando-se que são necessários seis pacotes para completar esse protocolo, alguém poderia perguntar por que não é utilizado um protocolo sem conexões. A resposta é que, em um mundo perfeito, esse tipo de protocolo poderia ser usado e, nesse caso, seriam necessários apenas dois pacotes: um para a solicitação e outro para a resposta. Entretanto, em face de mensagens extensas em qualquer sentido (por exemplo, um arquivo com vários megabytes), erros de transmissão e perda de pacotes, a situação se altera. Se a resposta consistisse em centenas de pacotes, alguns dos quais pudessem se perder durante a transmissão, como o cliente saberia que alguns fragmentos se perderam? Como o cliente saberia que o pacote último realmente recebido foi de fato o último pacote enviado? Suponha que o cliente quisesse um segundo arquivo. Como ele poderia distinguir o pacote 1 do segundo arquivo de um

pacote 1 perdido do primeiro arquivo que repentinamente tivesse encontrado o caminho até o cliente? Em resumo, no mundo real, um simples protocolo de solicitação/resposta sobre uma rede não confiável freqüentemente é inadequado. No Capítulo 3, estudaremos em detalhes uma variedade de protocolos que superam esses e outros problemas. Por enquanto, basta dizer que às vezes é muito conveniente ter um fluxo de bytes confiável e ordenado entre processos.

[T3] 1.3.5 O relacionamento entre serviços e protocolos

Serviços e protocolos são conceitos diferentes, embora sejam confundidos com freqüência. No entanto, essa distinção é tão importante que vamos enfatizá-la mais uma vez. Um *serviço* é um conjunto de primitivas (operações) que uma camada oferece à camada situada acima dela. O serviço define as operações que a camada está preparada para executar em nome de seus usuários, mas não informa absolutamente nada sobre como essas operações são implementadas. Um serviço se relaciona a uma interface entre duas camadas, sendo a camada inferior o fornecedor do serviço e a camada superior o usuário do serviço.

Já o *protocolo* é um conjunto de regras que controla o formato e o significado dos pacotes ou mensagens que são trocadas pelas entidades pares contidas em uma camada. As entidades utilizam protocolos com a finalidade de implementar suas definições de serviço. Elas têm a liberdade de trocar seus protocolos, desde que não alterem o serviço visível para seus usuários. Portanto, o serviço e o protocolo são independentes um do outro.

Em outras palavras, os serviços estão relacionados às interfaces entre camadas, como ilustra a Figura 1.19. Em contraste, os protocolos se relacionam aos pacotes enviados entre entidades pares de máquinas diferentes. É importante não confundir esses dois conceitos.

Vale a pena fazer uma analogia com as linguagens de programação. Um serviço é

um objeto ou um tipo de dados abstrato em uma linguagem orientada a objetos. Ele define as operações que podem ser executadas sobre um objeto, mas não especifica como essas operações são implementadas. Um protocolo se refere à *implementação* do serviço e, consequentemente, não é visto pelo usuário do serviço.

Em protocolos mais antigos, não havia muita distinção entre o serviço e o protocolo. Na prática, uma camada normal poderia ter uma primitiva de serviço SEND PACKET, com o usuário fornecendo um ponteiro para um pacote totalmente montado. Essa organização significava que todas as mudanças no protocolo ficavam imediatamente visíveis para os usuários. Hoje, a maioria dos projetistas de redes considera tal projeto um sério equívoco.

[arte: ver original p. 37]

[Dísticos]

[1]Camada k + 1

Camada k

Camada k – 1

[2]Serviço fornecido pela camada k

Protocolo

[3]Camada k + 1

Camada k

Camada k – 1

[F]Figura 1.19

[FL] O relacionamento entre um serviço e um protocolo

[T2]1.4 Modelos de referência

Depois de discutirmos em termos abstratos o conceito de redes divididas em camadas, vamos ver alguns exemplos práticos. Nas duas seções a seguir,

examinaremos duas importantes arquiteturas de rede, o modelo de referência OSI e o modelo de referência TCP/IP. Embora os *protocolos* associados ao modelo OSI raramente sejam usados nos dias de hoje, o *modelo* em si é de fato bastante geral e ainda válido, e as características descritas em cada camada ainda são muito importantes. O modelo TCP/IP tem características opostas: o modelo propriamente dito não é muito utilizado, mas os protocolos têm uso geral. Por essa razão, examinaremos ambos em detalhes. Além disso, às vezes é possível aprender mais com os fracassos do que com os sucessos.

[T3]1.4.1 O modelo de referência OSI

O modelo OSI (exceto o meio físico) é mostrado na Figura 1.20. Esse modelo se baseia em uma proposta desenvolvida pela ISO (International Standards Organization) como um primeiro passo em direção à padronização internacional dos protocolos empregados nas diversas camadas (Day e Zimmermann, 1983). Ele foi revisto em 1995 (Day, 1995). O modelo é chamado **Modelo de Referência ISO OSI (Open Systems Interconnection)**, pois ele trata da interconexão de sistemas abertos — ou seja, sistemas que estão abertos à comunicação com outros sistemas. Para abreviar, vamos denominá-lo simplesmente modelo OSI.

O modelo OSI tem sete camadas. Veja a seguir um resumo dos princípios aplicados para se chegar às sete camadas.

1. Uma camada deve ser criada onde houver necessidade de outro grau de abstração.
2. Cada camada deve executar uma função bem definida.
3. A função de cada camada deve ser escolhida tendo em vista a definição de protocolos padronizados internacionalmente.
4. Os limites de camadas devem ser escolhidos para minimizar o fluxo de informações pelas interfaces.

5. O número de camadas deve ser grande o bastante para que funções distintas não precisem ser desnecessariamente colocadas na mesma camada e pequeno o suficiente para que a arquitetura não se torne difícil de controlar.

Em seguida, discutiremos cada uma das camadas do modelo, começando pela camada inferior. Observe que o modelo OSI propriamente dito não é uma arquitetura de rede, pois não especifica os serviços e os protocolos exatos que devem ser usados em cada camada. Ele apenas informa o que cada camada deve fazer. No entanto, a ISO também produziu padrões para todas as camadas, embora esses padrões não façam parte do próprio modelo de referência. Cada um foi publicado como um padrão internacional distinto.

[T4] A camada física

A **camada física** trata da transmissão de bits brutos por um canal de comunicação. O projeto da rede deve garantir que, quando um lado enviar um bit 1, o outro lado o receberá como um bit 1, não como um bit 0. Nesse caso, as questões mais comuns são a voltagem a ser usada para representar um bit 1 e um bit 0, a quantidade de nanossegundos que um bit deve durar, o fato de a transmissão poder ser ou não realizada nos dois sentidos simultaneamente, a forma como a conexão inicial será estabelecida e de que maneira ela será encerrada quando ambos os lados tiverem terminado, e ainda quantos pinos o conector de rede terá e qual será a finalidade de cada pino. Nessa situação, as questões de projeto lidam em grande parte com interfaces mecânicas, elétricas e de sincronização, e com o meio físico de transmissão que se situa abaixo da camada física.

[arte: ver original p. 39]

[Dísticos]

[1]Camada

Protocolo de roteador/host da camada de enlace de dados

Protocolo de roteador/host da camada física

[5]Aplicação

Apresentação

Sessão

Transporte

Rede

Enlace de dados

Física

Host B

[6]Nome da unidade intercambiada

APDU

PPDU

SPDU

TPDU

Pacote

Quadro

Bit

[F]Figura 1.20

[FL] O modelo de referência OSI

[T4] A camada de enlace de dados

A principal tarefa da **camada de enlace de dados** é transformar um canal de transmissão bruta em uma linha que pareça livre de erros de transmissão não detectados para a camada de rede. Para executar essa tarefa, a camada de enlace de dados faz com que o transmissor divida os dados de entrada em **quadros de dados** (que, em geral, têm algumas centenas ou alguns milhares de bytes), e

transmita os quadros seqüencialmente. Se o serviço for confiável, o receptor confirmará a recepção correta de cada quadro, enviando de volta um **quadro de confirmação**.

Outra questão que surge na camada de enlace de dados (e na maioria das camadas mais altas) é como impedir que um transmissor rápido envie uma quantidade excessiva de dados a um receptor lento. Com freqüência, é necessário algum mecanismo que regule o tráfego para informar ao transmissor quanto espaço o buffer do receptor tem no momento. Muitas vezes, esse controle de fluxo e o tratamento de erros estão integrados.

As redes de difusão têm uma questão adicional a ser resolvida na camada de enlace de dados: como controlar o acesso ao canal compartilhado. Uma subcamada especial da camada de enlace de dados, a subcamada de controle de acesso ao meio, cuida desse problema.

[T4] A camada de rede

A **camada de rede** controla a operação da sub-rede. Uma questão fundamental de projeto é determinar a maneira como os pacotes são roteados da origem até o destino. As rotas podem se basear em tabelas estáticas, "amarradas" à rede e raramente alteradas. Elas também podem ser determinadas no início de cada conversação; por exemplo, uma sessão de terminal (como um logon em uma máquina remota). Por fim, elas podem ser altamente dinâmicas, sendo determinadas para cada pacote, com o objetivo de refletir a carga atual da rede.

Se houver muitos pacotes na sub-rede ao mesmo tempo, eles dividirão o mesmo caminho, provocando gargalos. O controle desse congestionamento também pertence à camada de rede. De modo mais geral, a qualidade do serviço fornecido (retardo, tempo em trânsito, instabilidade etc.) também é uma questão da camada de rede.

Quando um pacote tem de viajar de uma rede para outra até chegar a seu destino, podem surgir muitos problemas. O endereçamento utilizado pela segunda rede pode ser diferente do que é empregado pela primeira rede. Talvez a segunda rede não aceite o pacote devido a seu tamanho excessivo. Os protocolos podem ser diferentes e assim por diante. Cabe à camada de rede superar todos esses problemas, a fim de permitir que redes heterogêneas sejam interconectadas.

Nas redes de difusão, o problema de roteamento é simples, e assim a camada de rede com freqüência é estreita, ou mesmo inexistente.

[T4] A camada de transporte

A função básica da **camada de transporte** é aceitar dados da camada acima dela, dividi-los em unidades menores caso necessário, repassar essas unidades à camada de rede e assegurar que todos os fragmentos chegarão corretamente à outra extremidade. Além do mais, tudo isso deve ser feito com eficiência e de forma que as camadas superiores fiquem isoladas das inevitáveis mudanças na tecnologia de hardware.

A camada de transporte também determina que tipo de serviço deve ser fornecido à camada de sessão e, em última análise, aos usuários da rede. O tipo de conexão de transporte mais popular é um canal ponto a ponto livre de erros que entrega mensagens ou bytes na ordem em que eles foram enviados. No entanto, outros tipos possíveis de serviço de transporte são as mensagens isoladas sem nenhuma garantia relativa à ordem de entrega e à difusão de mensagens para muitos destinos. O tipo de serviço é determinado quando a conexão é estabelecida. (Observe que é impossível conseguir um canal livre de erros; o que as pessoas realmente entendem por essa expressão é que a taxa de erros é baixa o suficiente para ser ignorada na prática.)

A camada de transporte é uma verdadeira camada fim a fim, que liga a origem ao destino. Em outras palavras, um programa da máquina de origem mantém uma conversação com um programa semelhante instalado na máquina de destino, utilizando os cabeçalhos de mensagens e as mensagens de controle. Nas camadas inferiores, os protocolos são trocados entre cada uma das máquinas e seus vizinhos imediatos, e não entre as máquinas de origem e de destino, que podem estar separadas por muitos roteadores. A diferença entre as camadas de 1 a 3, que são encadeadas, e as camadas de 4 a 7, que são camadas fim a fim, é ilustrada na Figura 1.20.

[T4] A camada de sessão

A **camada de sessão** permite que os usuários de diferentes máquinas estabeleçam **sessões** entre eles. Uma sessão oferece diversos serviços, inclusive o **controle de diálogo** (mantendo o controle de quem deve transmitir em cada momento), o **gerenciamento de símbolos** (impedindo que duas partes tentem executar a mesma operação crítica ao mesmo tempo) e a **sincronização** (realizando a verificação periódica de transmissões longas para permitir que elas continuem a partir do ponto em que estavam ao ocorrer uma falha).

[T4] A camada de apresentação

Diferente das camadas mais baixas, que se preocupam principalmente com a movimentação de bits, a **camada de apresentação** está relacionada à sintaxe e à semântica das informações transmitidas. Para tornar possível a comunicação entre computadores com diferentes representações de dados, as estruturas de dados a serem intercambiadas podem ser definidas de maneira abstrata, juntamente com uma codificação padrão que será usada durante a conexão. A camada de apresentação gerencia essas estruturas de dados abstratas e permite a

definição e o intercâmbio de estruturas de dados de nível mais alto (por exemplo, registros bancários).

[T4] A camada de aplicação

A **camada de aplicação** contém uma série de protocolos comumente necessários para os usuários. Um protocolo de aplicação amplamente utilizado é o **HTTP (HyperText Transfer Protocol)**, que constitui a base para a World Wide Web. Quando um navegador deseja uma página da Web, ele envia o nome da página desejada ao servidor, utilizando o HTTP. Então, o servidor transmite a página de volta. Outros protocolos de aplicação são usados para transferências de arquivos, correio eletrônico e transmissão de notícias pela rede.

[T3] 1.4.2 O modelo de referência TCP/IP

Vamos deixar de lado o modelo de referência OSI e passar ao modelo de referência usado na "avó" de todas as redes de computadores geograficamente distribuídas, a ARPANET, e sua sucessora, a Internet mundial. Embora tenhamos deixado para depois a apresentação da história da ARPANET, será de grande utilidade entender alguns de seus principais aspectos. A ARPANET era uma rede de pesquisa patrocinada pelo Departamento de Defesa dos Estados Unidos (DoD). Pouco a pouco, centenas de universidades e repartições públicas foram conectadas, usando linhas telefônicas dedicadas. Quando foram criadas as redes de rádio e satélite, começaram a surgir problemas com os protocolos existentes, o que forçou a criação de uma nova arquitetura de referência. Desse modo, a habilidade para conectar várias redes de maneira uniforme foi um dos principais objetivos de projeto, desde o início. Mais tarde, essa arquitetura ficou conhecida como **Modelo de Referência TCP/IP**, graças a seus dois principais protocolos. Esse modelo foi definido pela primeira vez em (Cerf e Kahn, 1974). Uma nova

perspectiva foi oferecida mais tarde em (Leiner *et al.*, 1985). A filosofia de projeto na qual se baseia o modelo é discutida em (Clark, 1988).

Diante da preocupação do Departamento de Defesa dos EUA de que seus preciosos hosts, roteadores e gateways de interconexão de redes fossem destruídos de uma hora para outra, definiu-se também que a rede deveria ser capaz de sobreviver à perda do hardware de sub-redes, com as conversações existentes sendo mantidas em atividade. Em outras palavras, o Departamento de Defesa dos EUA queria que as conexões permanecessem intactas enquanto as máquinas de origem e de destino estivessem funcionando, mesmo que algumas máquinas ou linhas de transmissão intermediárias deixassem de operar repentinamente. Além disso, era necessária uma arquitetura flexível, capaz de se adaptar a aplicações com requisitos divergentes como, por exemplo, a transferência de arquivos e a transmissão de dados de voz em tempo real.

[T4] A camada inter-redes

Todas essas necessidades levaram à escolha de uma rede de comutação de pacotes baseada em um camada de interligação de redes sem conexões. Essa camada, chamada **camada inter-redes**, integra toda a arquitetura. Sua tarefa é permitir que os hosts injetem pacotes em qualquer rede e garantir que eles trafegarão independentemente até o destino (talvez em uma rede diferente). Eles podem chegar até mesmo em uma ordem diferente daquela em que foram enviados, obrigando as camadas superiores a reorganizá-los, caso a entrega em ordem seja desejável. Observe que, nesse caso, a expressão "inter-rede" é usada em sentido genérico, muito embora essa camada esteja presente na Internet.

A analogia usada nesse caso diz respeito ao sistema de correio (convencional). Uma pessoa pode deixar uma seqüência de cartas internacionais em uma caixa de correio em um país e, com um pouco de sorte, a maioria delas será entregue no

endereço correto no país de destino. Provavelmente, as cartas atravessarão um ou mais gateways internacionais ao longo do caminho, mas esse processo é transparente para os usuários. Além disso, o fato de cada país (ou seja, cada rede) ter seus próprios selos, tamanhos de envelope preferidos e regras de entrega fica oculto dos usuários.

A camada inter-redes define um formato de pacote oficial e um protocolo chamado **IP (Internet Protocol)**. A tarefa da camada inter-redes é entregar pacotes IP onde eles são necessários. O roteamento de pacotes é uma questão de grande importância nessa camada, assim como a necessidade de evitar o congestionamento. Por esses motivos, é razoável dizer que a função da camada inter-redes do TCP/IP é muito parecida com a da camada de rede do OSI. A Figura 1.21 mostra a correspondência entre elas.

[T4] A camada de transporte

No modelo TCP/IP, a camada localizada acima da camada inter-redes é chamada **camada de transporte**. A finalidade dessa camada é permitir que as entidades pares dos hosts de origem e de destino mantenham uma conversação, exatamente como acontece na camada de transporte OSI. Dois protocolos fim a fim foram definidos aqui. O primeiro deles, o **TCP (Transmission Control Protocol — protocolo de controle de transmissão)**, é um protocolo orientado a conexões confiável que permite a entrega sem erros de um fluxo de bytes originário de uma determinada máquina em qualquer computador da inter-rede. Esse protocolo fragmenta o fluxo de bytes de entrada em mensagens discretas e passa cada uma delas para a camada inter-redes. No destino, o processo TCP receptor volta a montar as mensagens recebidas no fluxo de saída. O TCP também cuida do controle de fluxo, impedindo que um transmissor rápido sobrecarregue um receptor lento com um volume de mensagens maior do que ele pode manipular.

[arte: ver original p. 43a]

[Dísticos]

[1]OSI

7 Aplicação

6 Apresentação

5 Sessão

4 Transporte

3 Rede

2 Enlace de dados

1 Física

[2]TCP/IP

Aplicação

Não presentes no modelo

Transporte

Inter-redes

Host/rede

[F]Figura 1.21

[FL]O modelo de referência TCP/IP

O segundo protocolo dessa camada, o **UDP (User Datagram Protocol — protocolo de datagrama do usuário)**, é um protocolo sem conexão não confiável para aplicações que não querem controle de fluxo nem manutenção da seqüência das mensagens enviadas, e desejam fornecer seus próprios recursos para isso. Ele também é amplamente utilizado em consultas e aplicações diretas do tipo cliente/servidor com solicitação/resposta, nas quais a entrega imediata é mais importante do que a entrega precisa, como a transmissão de dados de voz ou de vídeo. A relação entre o IP, o TCP e o UDP é mostrada na Figura 1.22. Desde que

o modelo foi desenvolvido, o IP foi implementado em muitas outras redes.

[arte: ver original p. 43b]

[Dísticos]

[1]Protocolos

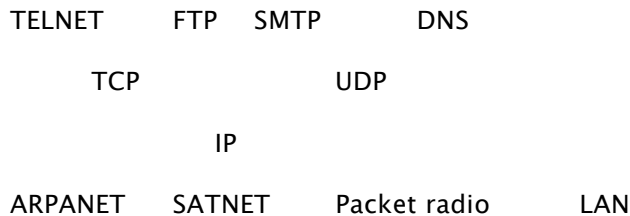
Redes

[2]

TELNET FTP SMTP DNS

TCP UDP

IP

ARPANET SATNET Packet radio LAN

[3]Camada (nomes no modelo OSI)

Aplicação

Transporte

Rede

Física + enlace de dados

[F]Figura 1.22

[FL] Protocolos e redes no modelo TCP/IP inicial

[T4] A camada de aplicação

O modelo TCP/IP não tem as camadas de sessão e de apresentação. Como não foi percebida qualquer necessidade, elas não foram incluídas. A experiência com o modelo OSI demonstrou a correção dessa tese: elas são pouco usadas na maioria das aplicações.

Acima da camada de transporte, encontramos a **camada de aplicação.** Ela contém todos os protocolos de nível mais alto. Dentre eles estão o protocolo de terminal virtual (TELNET), o protocolo de transferência de arquivos (FTP) e o protocolo de correio eletrônico (SMTP), como mostra a Figura 1.22. O protocolo de terminal

virtual permite que um usuário de um computador se conecte a uma máquina distante e trabalhe nela. O protocolo de transferência de arquivos permite mover dados com eficiência de uma máquina para outra. Originalmente, o correio eletrônico era um tipo de transferência de arquivos; no entanto, foi desenvolvido mais tarde um protocolo especializado para essa função (o SMTP). Muitos outros protocolos foram incluídos com o decorrer dos anos, como o DNS (Domain Name Service), que mapeia os nomes de hosts para seus respectivos endereços de rede, o NNTP, o protocolo usado para mover novos artigos de notícias da USENET, e o HTTP, o protocolo usado para buscar páginas na World Wide Web, entre muitos outros.

[T4] A camada host/rede

Abaixo da camada inter-redes, encontra-se um grande vácuo. O modelo de referência TCP/IP não especifica muito bem o que acontece ali, exceto o fato de que o host tem de se conectar à rede utilizando algum protocolo para que seja possível enviar pacotes IP. Esse protocolo não é definido e varia de host para host e de rede para rede. Os livros e a documentação que tratam do modelo TCP/IP raramente descrevem esse protocolo.

[T3] 1.4.3 Uma comparação entre os modelos de referência OSI e TCP/IP

Os modelos de referência OSI e TCP/IP têm muito em comum. Os dois se baseiam no conceito de uma pilha de protocolos independentes. Além disso, as camadas têm praticamente as mesmas funções. Por exemplo, em ambos os modelos estão presentes as camadas que englobam até a camada de transporte para oferecer um serviço de transporte fim a fim independente da rede a processos que desejam se comunicar. Essas camadas formam o provedor de transporte. Mais uma vez, em ambos os modelos, as camadas acima da camada de transporte

dizem respeito aos usuários orientados a aplicações do serviço de transporte.

Apesar dessas semelhanças fundamentais, os dois modelos também têm muitas diferenças. Nesta seção do livro, vamos nos deter nas principais diferenças existentes entre os dois modelos de referência. É importante notar que estamos comparando os *modelos de referência*, e não as *pilhas de protocolos* correspondentes. Os protocolos propriamente ditos serão discutidos em seguida. Para examinar as semelhanças e as diferenças entre o TCP/IP e o OSI, consulte (Piscitello e Chapin, 1993).

O modelo OSI tem três conceitos fundamentais:

1. Serviços
2. Interfaces
3. Protocolos

Provavelmente, a maior contribuição do modelo OSI seja tornar explícita a distinção entre esses três conceitos. Cada camada executa alguns serviços para a camada acima dela. A definição do *serviço* informa o que a camada faz, e não a forma como as entidades acima dela o acessam ou como a camada funciona. Essa definição estabelece a semântica da camada.

A *interface* de uma camada informa como os processos acima dela podem acessá-la. A interface especifica quais são os parâmetros e os resultados a serem esperados. Ela também não revela o funcionamento interno da camada.

Finalmente, os *protocolos* utilizados em uma camada são de responsabilidade dessa camada. A camada pode usar os protocolos que quiser, desde que eles viabilizem a realização do trabalho (ou seja, forneçam os serviços oferecidos). Ela também pode alterar esses protocolos sem influenciar o software das camadas superiores.

Essas idéias se adaptam perfeitamente aos novos conceitos da programação orientada a objetos. Um objeto, assim como uma camada, tem um conjunto de

métodos (operações) que os processos externos ao objeto podem invocar. A semântica desses métodos define o conjunto de serviços que o objeto oferece. Os parâmetros e os resultados dos métodos formam a interface do objeto. O código interno do objeto é seu protocolo, que não é visível nem interessa aos elementos que estão fora do objeto.

Originalmente, o modelo TCP/IP não distinguia com clareza a diferença entre serviço, interface e protocolo, embora as pessoas tenham tentado adaptá-lo ao modelo OSI. Por exemplo, os únicos serviços reais oferecidos pela camada inter-redes são SEND IP PACKET (enviar pacote IP) e RECEIVE IP PACKET (receber pacote IP).

Por essa razão, os protocolos do modelo OSI são mais bem encapsulados que os do modelo TCP/IP e podem ser substituídos com relativa facilidade, conforme as mudanças da tecnologia. Um dos principais objetivos das diversas camadas de protocolos é permitir a implementação dessas alterações.

O modelo de referência OSI foi concebido *antes* de os protocolos correspondentes terem sido criados. Isso significa que o modelo não foi desenvolvido com base em um determinado conjunto de protocolos, o que o deixou bastante flexível e genérico. No entanto, por não terem experiência no assunto, os projetistas não tinham muita noção sobre a funcionalidade que deveria ser incluída em cada camada.

Por exemplo, a camada de enlace de dados lidava originalmente com redes ponto a ponto. Quando surgiram as redes de difusão, foi preciso criar uma nova camada no modelo. Quando as pessoas começaram a criar redes reais com base no modelo OSI e nos protocolos existentes, elas perceberam que as especificações de serviço obrigatórias não eram compatíveis. Portanto, foi necessário enxertar no modelo subcamadas de convergência que permitissem atenuar as diferenças.

Por fim, como acreditava que cada país teria uma rede controlada pelo governo e

baseada nos protocolos OSI, o comitê não se preocupou com as conexões inter-redes. Para encurtar a história: na prática, tudo aconteceu de maneira muito diferente da teoria.

Com o TCP/IP, ocorreu exatamente o contrário: como os protocolos vieram primeiro, o modelo foi criado como uma descrição desses protocolos. Os protocolos não tiveram problemas para se adaptar ao modelo. Foi um casamento perfeito. O único problema foi o fato de o *modelo* não se adaptar a outras pilhas de protocolos. Conseqüentemente, ele não tinha muita utilidade para descrever outras redes que não faziam uso do protocolo TCP/IP.

Deixando a filosofia de lado e entrando em questões mais práticas, uma diferença óbvia entre os dois modelos está no número de camadas: o modelo OSI tem sete camadas e o TCP/IP tem quatro. Ambos têm as camadas de (inter-) rede, transporte e aplicação, mas as outras são diferentes.

Outra diferença está na área da comunicação sem conexão e da comunicação orientada a conexões. Na camada de rede, o modelo OSI é compatível com a comunicação sem conexão e com a comunicação orientada a conexões; no entanto, na camada de transporte, o modelo aceita apenas a comunicação orientada a conexões, onde ela de fato é mais importante (pois o serviço de transporte é visível para os usuários). O modelo TCP/IP só tem um modo de operação na camada de rede (sem conexão), mas aceita ambos os modos na camada de transporte, oferecendo aos usuários a possibilidade de escolha. Essa escolha é especialmente importante para os protocolos simples de solicitação/resposta.

[T3] 1.4.4 Uma crítica aos protocolos e ao modelo OSI

Nem o modelo OSI e seus respectivos protocolos nem o modelo TCP/IP e seus respectivos protocolos são perfeitos. Os dois têm sido alvo de uma série de

críticas. Nesta seção e na próxima, vamos examinar algumas delas; começaremos pelo modelo OSI e, em seguida, examinaremos o TCP/IP.

Na época em que a segunda edição americana deste livro foi publicada (1989), muitos especialistas tinham a impressão de que os protocolos e o modelo OSI controlariam o mundo e atropelariam tudo que se pusesse em seu caminho. Isso não aconteceu. Por quê? Vale a pena fazer uma revisão de algumas lições, que podem ser resumidas da seguinte maneira:

1. Momento ruim.
2. Tecnologia ruim.
3. Implementações ruins.
4. Política ruim.

[T4] Momento ruim

Vamos começar pelo problema mais importante: momento ruim. O momento em que um padrão é estabelecido é de fundamental importância para seu sucesso. David Clark, do M.I.T., tem uma teoria sobre os padrões que ele chama *apocalipse dos dois elefantes*, ilustrada na Figura 1.23.

[arte: ver original p. 47]

[Dísticos]

[1]Atividade

[2]Pesquisa

[3]Padrões

[4]Bilhões de dólares em investimentos

[5]Tempo

[F]Figura 1.23

[FL] O apocalipse dos dois elefantes

Essa figura mostra o volume de atividades relacionadas a um novo assunto. Quando o assunto é descoberto, há uma grande atividade de pesquisa na forma de discussões, artigos e reuniões. Após algum tempo dessa atividade inicial, as empresas descobrem o assunto e tem início a onda de bilhões de dólares em investimentos.

É essencial que os padrões sejam desenvolvidos entre os dois "elefantes". Se eles forem desenvolvidos muito cedo, antes de a pesquisa ser concluída, o assunto poderá não estar devidamente amadurecido e, conseqüentemente, surgirão padrões ruins. Se eles forem desenvolvidos muito tarde, muitas empresas talvez já tenham feito investimentos maciços para descobrir maneiras diferentes de tirar proveito dessa nova tecnologia e, portanto, os padrões serão efetivamente ignorados. Se o intervalo entre os dois elefantes for muito curto (porque todo mundo está ansioso para aproveitar as oportunidades anunciadas), a equipe de desenvolvimento dos padrões poderá se precipitar.

Hoje se sabe que o lançamento dos protocolos do padrão OSI foi precipitado. Os protocolos TCP/IP concorrentes já estavam sendo amplamente utilizados nas universidades de pesquisa na época em que apareceram os protocolos OSI. Antes mesmo do início da onda de investimentos de bilhões de dólares, o mercado acadêmico já era suficientemente grande, e muitos fabricantes começaram a oferecer produtos TCP/IP, apesar de estarem cautelosos. Quando surgiu o OSI, eles não estavam dispostos a investir em uma segunda pilha de protocolos enquanto ela não se tornasse uma imposição do mercado. Com todas as empresas aguardando que alguém desse o primeiro passo, o modelo OSI não saiu do papel.

[T4] Tecnologia ruim

A segunda razão para que o OSI não vingasse estava nas falhas do modelo e dos

protocolos. A escolha de sete camadas foi mais política do que técnica. Duas camadas (a de sessão e a de apresentação) estão praticamente vazias, enquanto duas outras (de enlace de dados e de rede) se encontram sobrecarregadas.

O modelo OSI, juntamente com os protocolos e as definições de serviços inter-relacionados, é extraordinariamente complexo. Quando empilhados, os padrões impressos chegam a quase um metro de altura. Além disso, eles são de difícil implementação e sua operação não é nada eficiente. Nesse contexto, vale a pena lembrar o enigma proposto por Paul Mockapetris e citado em (Rose, 1993):

P: O que você vê quando cruza com um mafioso que adota um padrão internacional?

R: Alguém que lhe faz uma oferta que você não pode entender.

Além de ser incompreensível, outro problema com o OSI é que algumas funções, como endereçamento, controle de fluxo e controle de erros, aparecem repetidamente em cada camada. Por exemplo, Saltzer *et a1.* (1984) lembraram que, para ser eficaz, o controle de erros deve ser feito na camada mais alta, de modo que sua repetição em cada uma das camadas inferiores seja desnecessária e ineficiente.

[T4] Implementações ruins

Devido à enorme complexidade do modelo e dos protocolos, ninguém ficou surpreso com o fato de as implementações iniciais serem lentas, pesadas e gigantescas. Todas as pessoas que as experimentaram, saíram chamuscadas. Não demorou muito para que elas associassem "OSI" a "baixa qualidade". A imagem resistiu inclusive às significativas melhorias a que os produtos foram submetidos com o decorrer do tempo.

Por outro lado, uma das primeiras implementações do TCP/IP fazia parte do UNIX de Berkeley e era muito boa (sem contar que era gratuita). As pessoas começaram

a usá-lo rapidamente, criando assim uma grande comunidade de usuários que, por sua vez, estimulou novas melhorias, que só serviram para aumentar ainda mais a base de usuários. Nesse caso, a espiral foi claramente ascendente.

[T4] Política ruim

Devido à implementação inicial, muitas pessoas, em particular no universo acadêmico, pensaram que o TCP/IP era parte do UNIX e, na década de 1980, as universidades tinham verdadeira adoração pelo UNIX.

Por outro lado, o OSI era considerado uma criação dos ministérios de telecomunicações europeus, da Comunidade Européia e, mais tarde, do governo dos Estados Unidos. Essa crença só era verdadeira em parte, mas a idéia de um punhado de burocratas tentando empurrar um padrão tecnicamente inferior pela garganta dos pobres pesquisadores e programadores que de fato trabalhavam no desenvolvimento de redes de computadores não foi de muita ajuda. Algumas pessoas viram nesse desenvolvimento uma repetição de um episódio da década de 1960, quando a IBM anunciou que a PL/I era a linguagem do futuro; mais tarde, essa afirmação foi desmentida pelo Departamento de Defesa dos EUA, que afirmou que a linguagem do futuro seria a Ada.

[T3] 1.4.5 Uma crítica ao modelo de referência TCP/IP

Os protocolos e o modelo TCP/IP também tiveram os seus problemas. Em primeiro lugar, o modelo não diferencia com a necessária clareza os conceitos de serviço, interface e protocolo. A boa prática da engenharia de software exige uma diferenciação entre especificação e implementação, algo que o OSI faz com muito cuidado, ao contrário do TCP/IP. Conseqüentemente, o modelo TCP/IP não é o melhor dos guias para a criação de novas redes com base em novas tecnologias.

Em segundo lugar, o modelo TCP/IP não é nem um pouco abrangente e não

consegue descrever outras pilhas de protocolos que não a pilha TCP/IP. Por exemplo, seria praticamente impossível tentar descrever a Bluetooth usando o modelo TCP/IP.

Em terceiro lugar, a camada host/rede não é realmente uma camada no sentido em que o termo é usado no contexto dos protocolos hierarquizados. Trata-se, na verdade, de uma interface (entre as camadas de rede e de enlace de dados). A distinção entre uma interface e uma camada é crucial e você deve considerá-la com cuidado.

Em quarto lugar, o modelo TCP/IP não faz distinção (nem sequer menciona) entre as camadas física e de enlace de dados. Elas são completamente diferentes. A camada física está relacionada às características de transmissão do fio de cobre, dos cabos de fibra óptica e da comunicação sem fio. A tarefa da camada de enlace de dados é delimitar o início e o final dos quadros e enviá-los de um lado a outro com o grau de confiabilidade desejado. Um modelo mais adequado deve incluir as duas camadas como elementos distintos. O modelo TCP/IP não faz isso.

Por fim, apesar de os protocolos IP e TCP terem sido cuidadosamente projetados e bem implementados, o mesmo não aconteceu com muitos outros protocolos produzidos pela comunidade acadêmica. As implementações desses protocolos eram distribuídas gratuitamente, o que acabava difundindo seu uso de tal forma que se tornou difícil substituí-las. Hoje em dia, a fidelidade a esse produtos é motivo de alguns embaraços. Por exemplo, o protocolo de terminal virtual, o TELNET, foi projetado para um terminal TTY mecânico, capaz de processar 10 caracteres por segundo. Ele não reconhece o mouse e as interfaces gráficas do usuário. No entanto, esse protocolo é usado em larga escala ainda hoje, 25 anos depois de seu surgimento.

Em resumo, apesar de seus problemas, o *modelo* OSI (sem as camadas de sessão e apresentação) mostrou-se excepcionalmente útil para a discussão das redes de

computadores. Por outro lado, os *protocolos* OSI jamais conseguiram se tornar populares. Ocorre exatamente o contrário com o TCP/IP: o *modelo* é praticamente inexistente, mas os *protocolos* são usados em larga escala. Como os cientistas da computação gostam de valorizar aquilo que eles próprios desenvolvem, usaremos neste livro um modelo OSI modificado, mas nos concentraremos basicamente no TCP/IP e em protocolos afins, bem como em recursos mais modernos, como 802, SONET e Bluetooth. Na verdade, a estrutura básica deste livro utilizará o modelo híbrido da Figura 1.24.

[arte: ver original p. 49]

[Dísticos]

[1]5 Camada de aplicação

[2]4 Camada de transporte

[3]3 Camada de rede

[4]2 Camada de enlace de dados

[5]1 Camada física

[F]Figura 1.24

[FL] O modelo de referência híbrido que será usado neste livro

[T2] 1.5 Exemplos de redes

O assunto de redes de computadores abrange muitos tipos diferentes de redes, grandes e pequenas, bem conhecidas e pouco conhecidas. Elas têm diferentes objetivos, escalas e tecnologias. Nas seções a seguir, examinaremos alguns exemplos, para termos uma idéia da variedade existente na área de redes de computadores.

Começaremos com a Internet, talvez a rede mais conhecida, e estudaremos sua história, sua evolução e sua tecnologia. Em seguida, consideraremos a ATM, utilizada com freqüência no núcleo de grandes redes (de telefonia). Tecnicamente

ela é muito diferente da Internet, havendo um bom contraste entre ambas.

Depois, apresentaremos a Ethernet, a rede local dominante no mercado. Por fim, veremos o IEEE 802.11, o padrão para LANs sem fios.

[T3] 1.5.1 A Internet

A Internet não é de modo algum uma rede, mas sim um vasto conjunto de redes diferentes que utilizam certos protocolos comuns e fornecem determinados serviços comuns. É um sistema pouco usual no sentido de não ter sido planejado nem ser controlado por ninguém. Para entendê-la melhor, vamos começar do início e observar como e por que ela foi desenvolvida. Se desejar conhecer uma história maravilhosa sobre o surgimento da Internet, recomendamos o livro de John Naughton (2000). Trata-se de um daqueles raros livros que não apenas são interessantes, mas que também tem 20 páginas de citações destinadas aos historiadores sérios. Uma parte do material a seguir se baseia nesse livro.
É claro que também foram escritos incontáveis livros técnicos sobre a Internet e seus protocolos. Para obter mais informações consulte, por exemplo, (Maufer, 1999).

[T4] A ARPANET

A história começa no final da década de 1950. No auge da Guerra Fria, o Departamento de Defesa dos EUA queria uma rede de controle e comando capaz de sobreviver a uma guerra nuclear. Nessa época, todas as comunicações militares passavam pela rede de telefonia pública, considerada vulnerável. A razão para essa convicção pode ser vista na Figura 1.25(a). Nessa figura, os pontos pretos representam centrais de comutação telefônica, cada uma das quais conectada a milhares de telefones. Por sua vez, essas centrais de comutação estavam conectadas a centrais de comutação de nível mais alto (centrais

interurbanas), formando uma hierarquia nacional apenas com uma pequena redundância. A vulnerabilidade do sistema era o fato de que a destruição de algumas centrais interurbanas importantes poderia fragmentar o sistema em muitas ilhas isoladas.

Por volta de 1960, o Departamento de Defesa dos EUA firmou um contrato com a RAND Corporation para encontrar uma solução. Um de seus funcionários, Paul Baran, apresentou o projeto altamente distribuído e tolerante a falhas da Figura 1.25(b). Tendo em vista que os caminhos entre duas centrais de comutação quaisquer eram agora muito mais longos do que a distância que os sinais análogos podiam percorrer sem distorção, Baran propôs o uso da tecnologia digital de comutação de pacotes em todo o sistema. Baran enviou diversos relatórios para o Departamento de Defesa dos EUA descrevendo suas idéias em detalhes. Os funcionários do Pentágono gostaram do conceito e pediram à AT&T, na época a empresa que detinha o monopólio nacional da telefonia nos Estados Unidos, que construísse um protótipo. A AT&T descartou as idéias de Baran. Afinal, a maior e mais rica corporação do mundo não podia permitir que um jovem pretensioso lhe ensinasse a criar um sistema telefônico. A empresa informou que a rede de Baran não podia ser construída, e a idéia foi abandonada.

[arte: ver original p. 51]

[Dísticos]

[1]Central de comutação

[2] Central interurbana

[3](a) (b)

[F]Figura 1.25

[FL] (a) Estrutura do sistema de telefonia. (b) Sistema distribuído de comutação proposto por Baran

Vários anos se passaram e o Departamento de Defesa dos EUA ainda não tinha um sistema melhor de comando e controle. Para entender o que aconteceu em seguida, temos de retornar a outubro de 1957, quando a União Soviética bateu os Estados Unidos na corrida espacial com o lançamento do primeiro satélite artificial, o Sputnik. Quando tentou descobrir quem tinha "dormido no ponto", o Presidente Eisenhower acabou detectando a disputa entre o Exército, a Marinha e a Força Aérea pelo orçamento de pesquisa do Pentágono. Sua resposta imediata foi criar uma única organização de pesquisa de defesa, a **ARPA**, ou **Advanced Research Projetcts Agency**. A ARPA não tinha cientistas nem laboratórios; de fato, ela não tinha nada além de um escritório e de um pequeno orçamento (pelos padrões do Pentágono). A agência realiza seu trabalho oferecendo concessões e contratos a universidades e empresas cujas idéias lhe pareciam promissoras. Durante os primeiros anos, a ARPA tentou compreender qual deveria ser sua missão mas, em 1967, a atenção do então diretor da ARPA, Larry Roberts, se voltou para as redes. Ele entrou em contato com diversos especialistas para decidir o que fazer. Um deles, Wesley Clark, sugeriu a criação de uma sub-rede comutada por pacotes, dando a cada host seu próprio roteador, como ilustra a Figura 1.10.

Após algum ceticismo inicial, Roberts comprou a idéia e apresentou um documento um tanto vago sobre ela no ACM SIGOPS Symposium on Operating System Principles, realizado em Gatlinburg, Tennessee, no final de 1967 (Roberts, 1967). Para grande surpresa de Roberts, outro documento na conferência descrevia um sistema semelhante, que não apenas tinha sido projetado mas, na realidade, havia sido implementado sob a orientação de Donald Davies do National Physical Laboratory, na Inglaterra. O sistema do NPL não era um sistema nacional (ele simplesmente conectava vários computadores no campus do NPL), mas demonstrava que a comutação de pacotes podia funcionar. Além disso, ele

citava o trabalho anteriormente descartado de Baran. Roberts voltou de Gatlinburg determinado a construir o que mais tarde ficou conhecido como **ARPANET.**

A sub-rede consistiria em minicomputadores chamados **IMPs (Interface Message Processors — processadores de mensagens de interface)** conectados por linhas de transmissão de 56 kbps. Para garantir sua alta confiabilidade, cada IMP seria conectado a pelo menos dois outros IMPs. A sub-rede tinha de ser uma sub-rede de datagrama, de modo que, se algumas linhas e alguns IMPs fossem destruídos, as mensagens pudessem ser roteadas automaticamente para caminhos alternativos.

Cada nó da rede deveria ter um IMP e um host na mesma sala, conectados por um fio curto. Um host poderia enviar mensagens de até 8063 bits para seu IMP que, em seguida, dividiria essas mensagens em pacotes de no máximo 1008 bits e os encaminharia de forma independente até o destino. Cada pacote era recebido integralmente antes de ser encaminhado; assim, a sub-rede se tornou a primeira rede eletrônica de comutação de pacotes de store-and-forward (de armazenamento e encaminhamento).

Em seguida, a ARPA abriu uma concorrência para a construção da sub-rede. Doze empresas apresentaram propostas. Depois de avaliar todas as propostas, a ARPA selecionou a BBN, uma empresa de consultoria de Cambridge, Massachusetts e, em dezembro de 1968, assinou um contrato para montar a sub-rede e desenvolver o software para ela. A BBN resolveu utilizar como IMPs minicomputadores Honeywell DDP-316 especialmente modificados, com 12 K palavras de 16 bits de memória principal. Os IMPs não tinham unidades de discos, pois os componentes móveis eram considerados pouco confiáveis. Os IMPs eram interconectados por linhas privadas de 56 kbps das companhias telefônicas. Embora 56 kbps seja agora a única escolha de adolescentes que não

podem dispor de ADSL ou modems a cabo, na época era o melhor que o dinheiro podia comprar.

O software foi dividido em duas partes: sub-rede e host. O software da sub-rede consistia na extremidade IMP da conexão host-IMP, no protocolo IMP-IMP e em um protocolo do IMP de origem para o IMP de destino, criado para aumentar a confiabilidade. O projeto original da ARPANET é mostrado na Figura 1.26.

[arte: ver original p. 52]

[Dísticos]

[1]Protocolo host-IMP

[2]Protocolo host-host

[3]Host

[4]Protocolo de IMP de origem para IMP de destino

[5]Protocolo IMP-IMP

[6]Protocolo IMP-IMP

[7]Sub-rede

[8]IMP

[F]Figura 1.26

[FL] O projeto original da ARPANET

Fora da sub-rede, também havia necessidade de software, ou seja, da extremidade referente ao host na conexão host-IMP, do protocolo host-host e do software de aplicação. A BBN logo percebeu que tinha realizado seu compromisso a contento quando enviou uma mensagem em uma ligação host-IMP.

Roberts tinha um problema: os hosts também precisavam de software. Para lidar com ele, Roberts convocou uma reunião com os pesquisadores de rede que, em sua maioria, era formada por estudantes universitários, em Snowbird, Utah, no verão de 1969. Os universitários esperavam que algum perito em redes

explicasse o projeto geral da rede e seu software, e depois atribuísse a cada um deles a tarefa de desenvolver uma parte do projeto. Eles ficaram absolutamente surpresos ao verem que não havia nenhum especialista em rede e nenhum projeto geral. Os estudantes tinham de entender o trabalho que estavam prestes a realizar.

No entanto, em dezembro de 1969 entrou no ar uma rede experimental com quatro nós (UCLA, UCSB, SRI e University of Utah). Esses quatro nós foram escolhidos porque todos tinham um grande número de contratos com a ARPA, e todos tinham computadores host diferentes e completamente incompatíveis (o que dava um certo charme ao desafio). A rede cresceu rapidamente à medida que outros IMPs foram entregues e instalados; logo se estendeu por todo o território norte-americano. A Figura 1.27 mostra a rapidez com que a ARPANET se desenvolveu nos três primeiros anos.

[arte: ver original p. 53]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 1.27

[FL] O crescimento da ARPANET. (a) Dezembro de 1969. (b) Julho de 1970. (c) Março de 1971. (d) Abril de 1972. (e) Setembro de 1972

Além de ajudar o súbito crescimento da ARPANET, a ARPA também financiou pesquisas sobre o uso de redes de satélite e redes móveis de rádio de pacotes. Em uma famosa demonstração, um motorista de caminhão viajando pela Califórnia utilizou a rede de rádio de pacotes para enviar mensagens à SRI, que foram então encaminhadas pela ARPANET até a Costa Leste dos EUA, de onde foram enviadas à University College, em Londres, pela rede de satélite. Isso

permitiu que um pesquisador no caminhão usasse um computador situado em Londres enquanto dirigia pelo estado da Califórnia.

Essa experiência também demonstrou que os protocolos da ARPANET não eram adequados para execução em várias redes. Essa observação levou a mais pesquisas sobre protocolos, culminando com a invenção dos protocolos e do modelo TCP/IP (Cerf e Kahn, 1974). O TCP/IP foi criado especificamente para manipular a comunicação sobre inter-redes, algo que se tornou mais importante à medida que um número maior de redes era conectado à ARPANET.

Para estimular a adoção desses novos protocolos, a ARPA ofereceu diversos contratos à BBN e à University of California, em Berkeley para integrá-los no UNIX de Berkeley. Os pesquisadores de Berkeley desenvolveram uma interface de programa conveniente para a rede (soquetes) e criaram muitos programas aplicativos, utilitários e de gerenciamento para facilitar a interligação de redes.

A ocasião não podia ser mais propícia. Muitas universidades tinham acabado de adquirir um segundo ou um terceiro computador VAX e uma LAN para conectá-los, mas não tinham nenhum software de rede. Quando surgiu o 4.2BSD, com TCP/P, soquetes e muitos utilitários de rede, o pacote completo foi adotado imediatamente. Além disso, com o TCP/IP, era fácil conectar as LANs à ARPANET, o que efetivamente acabou acontecendo.

Durante a década de 1980, novas redes, em particular LANs, foram conectadas à ARPANET. À medida que a escala aumentou, tornou-se cada vez mais dispendioso localizar hosts, e assim foi criado o **DNS (Domain Naming System)**, cujo objetivo era organizar máquinas em domínios e mapear nomes de hosts em endereços IP. Desde então, o DNS se transformou em um sistema generalizado de bancos de dados distribuídos, capaz de armazenar uma série de informações referentes à atribuição de nomes. Estudaremos esse sistema em detalhes no Capítulo 7.

[T4] NSFNET

No final da década de 1970, a NSF (National Science Foundation) percebeu o enorme impacto que a ARPANET estava causando nas pesquisas universitárias nos Estados Unidos, permitindo que cientistas de todo o país compartilhassem dados e trabalhassem juntos em projetos de pesquisa. No entanto, para entrar na ARPANET, uma universidade precisava ter um contrato de pesquisa com o Departamento de Defesa dos EUA, privilégio que muitas não tinham. A resposta da NSF foi desenvolver uma sucessora para a ARPANET, que seria aberta a todos os grupos de pesquisa universitários. Para ter algo concreto com que começar, a NSF decidiu construir uma rede de backbone para conectar seus seis centros de supercomputadores, localizados em San Diego, Boulder, Champaign, Pittsburgh, Ithaca e Princeton. Cada supercomputador ganhou um irmão caçula, um microcomputador LSI-11, chamado **fuzzball**. Os fuzzballs estavam conectados a linhas privadas de 56 Kbps e formavam a sub-rede, usando a mesma tecnologia de hardware da ARPANET. Porém, a tecnologia de software era diferente: os fuzzballs se comunicavam diretamente com o TCP/IP desde o início, criando assim a primeira WAN TCP/IP.

A NSF também financiou cerca de 20 redes regionais que foram conectadas ao backbone para que os usuários de milhares de universidades, laboratórios de pesquisa, bibliotecas e museus tivessem acesso a um dos supercomputadores e se comunicassem entre si. A rede completa, incluindo o backbone e as redes regionais, foi chamada **NSFNET**. Ela se conectava à ARPANET por meio de um link entre um IMP e um fuzzball na central de processamento de dados de Carnegie-Mellon. O primeiro backbone da NSFNET está ilustrado na Figura 1.28.

[arte: ver original p. 55]

[Dísticos]

[1]Centro de supercomputadores da NSF

[2]Rede de nível intermediário da NSF

[3]Ambos

[F]Figura 1.28

[FL] O backbone da NSFNET em 1988

A NSFNET foi um sucesso instantâneo e logo estava sobrecarregada. Imediatamente, a NSF começou a planejar sua sucessora e firmou um contrato com o consórcio MERIT de Michigan para executá-la. Junto à MCI foram alugados canais de fibra óptica de 448 Kbps (na época absorvida pela WorldCom) para fornecer a versão 2 do backbone. Máquinas IBM PC-RT foram usadas como roteadores. Logo, o segundo backbone também estava operando com sua capacidade máxima e, em 1990, ele foi atualizado para 1,5 Mbps.

O contínuo crescimento levou a NSF a perceber que o governo não podia continuar a financiar a rede para sempre. Além disso, as organizações comerciais queriam participar da rede, mas eram proibidas pelo estatuto da NSF de utilizar redes mantidas com verbas da fundação. Conseqüentemente, a NSF estimulou a MERIT, a MCI e a IBM a formarem uma empresa sem fins lucrativos, a **ANS (Advanced Networks and Services)** o que, na prática, foi a primeira etapa em direção à comercialização. Em 1990, a ANS assumiu a NSFNET e atualizou os links de 1,5 Mbps para 45 Mbps, a fim de formar a **ANSNET**. Essa rede operou por 5 anos e depois foi vendida à América Online. Porém, nessa época, diversas empresas estavam oferecendo o serviço IP comercial e se tornou claro que o governo deveria deixar o negócio de redes.

Para facilitar a transição e garantir que todas as redes regionais pudessem se comunicar entre si, a NSF contratou quatro diferentes operadoras de redes para estabelecer um **NAP (Network Access Point — ponto de acesso de rede).** Essas

operadoras eram a PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.) e Sprint (cidade de Nova York onde, para fins de NAP, a localidade de Pennsauken, em Nova Jersey, pertence à cidade de Nova York). Todas as operadoras de redes que quisessem oferecer serviços de backbone às redes regionais da NSF tinham de estabelecer conexão com todos os NAPs.

Nessa estratégia, um pacote originário de uma das redes regionais tinha a opção de escolher uma das concessionárias de backbone para ser transferido do NAP de origem para o NAP de destino. Conseqüentemente, as concessionárias de backbone foram obrigadas a concorrer com as redes regionais, tendo de oferecer preços e serviços melhores para se manterem no mercado. Como resultado, o conceito de um único backbone padrão foi substituído por uma infra-estrutura competitiva, com fins lucrativos. Muitas pessoas gostam de criticar o governo dos Estados Unidos por não ser inovador mas, na área de redes, foram o Departamento de Defesa dos EUA e a NSF que criaram a infra-estrutura que formou a base para a Internet, e depois a entregaram à indústria para cuidar de sua operação.

Durante a década de 1990, muitos outros países e regiões também construíram redes nacionais de pesquisa, com freqüência moldadas de acordo com a ARPANET e a NSFNET. Na Europa, essas redes incluíram EuropaNET e EBONE, que começaram com linhas de 2 Mbps e depois foram atualizadas com linhas de 34 Mbps. Mais tarde, a infra-estrutura de rede na Europa também foi entregue à indústria.

[T4] Utilização da Internet

O número de redes, máquinas e usuários conectados à ARPANET cresceu rapidamente depois que o TCP/IP se tornou o único protocolo oficial, em 1º de janeiro de 1983. Quando a NSFNET e a ARPANET foram interconectadas, o

crescimento tornou-se exponencial. Muitas redes regionais foram integradas, e foram criadas conexões com redes no Canadá, na Europa e no Pacífico.

Em meados da década de 1980, as pessoas começaram a ver um conjunto de redes como uma inter-rede e, mais tarde, como a Internet, apesar de não ter havido uma cerimônia oficial com políticos quebrando uma garrafa de champanhe em um fuzzball.

Os elementos que formam a base da Internet são o modelo de referência TCP/IP e a pilha de protocolos TCP/IP. O TCP/IP possibilita a criação de um serviço universal e pode ser comparado ao sistema telefônico e à adoção da bitola padrão pelas ferrovias no Século XIX, ou com a adoção de protocolos comuns de sinalização por todas as companhia telefônicas.

Então, o que significa estar na Internet? Nossa definição é a de que uma máquina está na Internet quando executa a pilha de protocolos TCP/IP, tem um endereço IP e pode enviar pacotes IP a todas as outras máquinas da Internet. A capacidade de enviar e receber mensagens de correio eletrônico não é suficiente, pois o correio eletrônico é encaminhado por gateway a muitas redes que estão fora da Internet. No entanto, a questão fica um pouco nebulosa pelo fato de milhões de computadores pessoais poderem acessar um provedor de serviços da Internet utilizando um modem, receberem a atribuição de um endereço IP temporário e enviarem pacotes IP a outros hosts da Internet. Na verdade, essas máquinas também pertencem à Internet, já que estão conectadas ao roteador do provedor de serviços.

Tradicionalmente (o que significa de 1970 a cerca de 1990), a Internet e suas predecessoras tinham quatro aplicações principais:

1. **Correio eletrônico (e-mail).** A possibilidade de redigir, enviar e receber mensagens de correio eletrônico é uma realidade criada já na fase inicial da ARPANET e é imensamente popular. Muitas pessoas recebem dezenas de

mensagens por dia e fazem do correio eletrônico sua principal forma de interação com o mundo exterior, usando-o com muito mais freqüência do que o telefone e o correio tradicionais. Atualmente, os programas de correio eletrônico estão disponíveis em quase todos os tipos de computadores.

2. **Newsgroups.** Os newsgroups são fóruns especializados, nos quais usuários com interesses comuns podem trocar mensagens. Existem milhares de newsgroups, dedicados a tópicos técnicos e não técnicos, inclusive computadores, ciência, lazer e política. Cada newsgroup tem sua própria etiqueta (regras para utilização do serviço), seu estilo e seus costumes; as pessoas que os violam podem até ser expulsas.

3. **Logon remoto.** Utilizando os programas telnet, rlogin ou ssh, os usuários de qualquer lugar na Internet podem se conectar a qualquer outra máquina na qual tenham uma conta.

4. **Transferência de arquivos.** Utilizando o programa FTP, é possível copiar arquivos entre máquinas ligadas à Internet. Dessa forma, você pode ter acesso a inúmeros artigos, bancos de dados e outras informações.

Até o início da década de 1990, a Internet era um verdadeiro reduto de pesquisadores ligados às universidades, ao governo e à indústria. Uma nova aplicação, a **WWW (World Wide Web)**, mudou essa realidade e atraiu para a rede milhares de novos usuários, sem a menor pretensão acadêmica. Essa aplicação, criado pelo físico da CERN Tim Berners-Lee, facilitou sobremaneira seu uso, muito embora não tenha alterado os recursos oferecidos pela rede. Junto com o navegador Mosaic, desenvolvido por Marc Andreessen no NCSA (National Center for Supercomputer Applications) em Urbana, Illinois, a WWW tornou possível a configuração de diversas páginas de informações de um site contendo texto, figuras, sons e até mesmo vídeo, com links incorporados para outras páginas. Clicando em um link, o usuário é repentinamente transportado para a página

indicada por esse link. Por exemplo, muitas empresas têm uma home page com entradas que remetem a outras páginas contendo informações sobre seus produtos, listas de preços, vendas, suporte técnico, comunicação com funcionários, informações para acionistas e muito mais.

Foram criados muitos outros tipos de páginas em um período de tempo muito curto, incluindo mapas, indicadores financeiros, catálogos de fichas de biblioteca, programas de rádio gravados e até mesmo uma página apontando para o texto completo de muitos livros cujos direitos autorais caíram em domínio público (Mark Twain, Charles Dickens etc.). Muitas pessoas também têm páginas pessoais (as chamadas home pages pessoais).

Grande parte desse crescimento durante a década de 1990 foi impulsionado por empresas denominadas **provedores de serviços da Internet** (ISPs – Internet Service Providers). Essas empresas oferecem a usuários individuais a possibilidade de acessar uma de suas máquinas e se conectar à Internet, obtendo assim acesso ao correio eletrônico, à WWW e a outros serviços da Internet. Essas empresas reuniram dezenas de milhões de novos usuários por ano durante a década passada, alterando completamente a característica da rede, que passou de um jogo acadêmico e militar para um serviço de utilidade pública, muito semelhante ao sistema telefônico. O número de usuários da Internet é desconhecido no momento, mas sem dúvida chega a centenas de milhões em todo o mundo e provavelmente alcançará em breve 1 bilhão de pessoas.

[T4] Arquitetura da Internet

Nesta seção, tentaremos apresentar uma breve visão geral da Internet atual. Devido às muitas fusões entre empresas de telefonia e ISPs, as águas ficaram turvas e muitas vezes é difícil saber quem está fazendo o que.

Conseqüentemente, esta descrição terá de ser um pouco mais simples que a

realidade. O quadro geral é mostrado na Figura 1.29. Vamos agora examinar cada item dessa figura.

[arte: ver original p. 58]

[Dísticos]

[1]ISP regional

[2]Backbone

[3]POP

[4]NAP

[5]Grupo de servidores

[6]Roteador

[7]LAN corporativa

[8]Cliente

[9]Sistema telefônico

[F]Figura 1.29

[FL] Visão geral da Internet

Um bom lugar para começar é a casa de um cliente. Vamos supor que nosso cliente acesse seu ISP usando uma linha telefônica de discagem, como mostra a Figura 1.29. O modem é uma placa dentro do PC que converte os sinais digitais que o computador produz em sinais análogos que podem passar livremente pelo sistema telefônico. Esses sinais são transferidos para o **POP (Point of Presence — ponto de presença)** do ISP, onde são removidos do sistema telefônicos e injetados na rede regional do ISP. Desse ponto em diante, o sistema é totalmente digital e comutado por pacotes. Se o ISP for a empresa de telefonia local, o POP provavelmente estará localizado na central de comutação telefônica onde termina a fiação telefônica do cliente. Se o ISP não for a empresa de telefonia local, o POP poderá estar algumas centrais de comutação mais afastado.

A rede regional do ISP consiste em roteadores interconectados nas várias cidades servidas pelo ISP. Se o pacote se destinar a um host servido diretamente pelo ISP, ele será entregue ao host. Caso contrário, o pacote será encaminhado à operadora de backbone do ISP.

No nível superior, estão as operadoras de backbone importantes, empresas como AT&T e Sprint. Elas operam grandes redes internacionais de backbones, com milhares de roteadores conectados por fibra óptica de alta largura de banda. Grandes corporações e serviços de hosts que controlam grupos de servidores (server farms, máquinas que podem servir milhares de páginas da Web por segundo) muitas vezes se conectam diretamente ao backbone. As operadoras de backbones incentivam essa conexão direta, alugando espaço nos chamados **hotéis de concessionárias**, basicamente bastidores (racks) de equipamentos na mesma sala em que está o roteador, a fim de permitir conexões curtas e rápidas entre os grupos de servidores e o backbone.

Se um pacote entregue ao backbone se destinar ao ISP ou a uma empresa servida pela backbone, ele será enviado ao roteador mais próximo e entregue. Porém, existem no mundo muitos backbones de diversos tamanhos, e assim um pacote talvez tenha de passar a um backbone concorrente. Para permitir que os pacotes saltem entre os backbones, todos os backbones importantes se conectam aos NAPs descritos antes. Basicamente, um NAP é uma sala repleta de roteadores, pelo menos um por backbone. Uma LAN na sala conecta todos os roteadores, de forma que os pacotes possam ser encaminhados de qualquer backbone para qualquer outro. Além de estarem interconectados a NAPs, os backbones maiores têm numerosas conexões diretas entre seus roteadores, uma técnica conhecida como **@@@formação de pares privados**. Um dos muitos paradoxos da Internet é o fato de ISPs que concorrem publicamente uns com os outros pelos clientes muitas vezes colaborarem de forma reservada na formação de pares privados (Metz,

2001).

Isso encerra nosso rápido tour pela Internet. Teremos muito a dizer sobre os componentes individuais e seu projeto, seus algoritmos e seus protocolos em capítulos posteriores. Também vale a pena mencionar de passagem que algumas empresas interconectam todas as suas redes internas, freqüentemente usando a mesma tecnologia da Internet. Essas **intranets** em geral só estão acessíveis dentro da empresa mas, em todos os outros aspectos, funcionam do mesmo modo que a Internet.

[T3] 1.5.2 Redes orientadas a conexões: X.25, Frame Relay e ATM

Desde o início das redes, há uma guerra entre as pessoas que admitem sub-redes sem conexões (isto é, datagramas) e as pessoas que admitem sub-redes orientadas a conexões. Os principais proponentes das sub-redes sem conexões vêm da comunidade da ARPANET/Internet. Lembre-se de que o desejo original do Departamento de Defesa dos EUA ao financiar e construir a ARPANET era ter uma rede que continuasse a funcionar mesmo depois de vários ataques diretos por armas nucleares que destruíssem numerosos roteadores e linhas de transmissão. Desse modo, a tolerância a falhas estava no topo de sua lista de prioridades, mas não a cobrança aos clientes. Essa abordagem resultou em um projeto sem conexões, no qual cada pacote é roteado de modo independente de qualquer outro pacote. Em conseqüência disso, se alguns roteadores ficarem inativos durante uma sessão, não haverá nenhum dano, desde que o sistema possa se reconfigurar dinamicamente, para que os pacotes subseqüentes possam encontrar alguma rota até o destino, mesmo que ela seja diferente da que foi utilizada por pacotes anteriores.

O das redes orientadas a conexões vem do mundo das empresas de telefonia. No sistema de telefonia, um visitante deve discar o número do telefone chamado e

esperar por uma conexão antes de falar ou enviar os dados. Essa configuração de conexão estabelece uma rota pelo sistema de telefonia que é mantida até a chamada ser encerrada. Todas as palavras ou pacotes seguem a mesma rota. Se uma linha ou um switch no caminho sofrer uma pane, a chamada será cancelada. Era exatamente essa propriedade que desagradava ao Departamento de Defesa dos EUA.

Então, por que as empresas de telefonia apreciam esse sistema? Há duas razões:

1. Qualidade de serviço.
2. Faturamento.

Configurando uma conexão com antecedência, a sub-rede pode reservar recursos como espaço de buffer e capacidade da CPU do roteador. Se houver uma tentativa de configurar uma chamada e não houver recursos suficientes disponíveis, a chamada será rejeitada e o chamador receberá uma espécie de sinal de ocupado. Desse modo, uma vez estabelecida, a conexão receberá um bom serviço. Com uma rede sem conexões, se um número excessivo de pacotes chegar ao mesmo roteador no mesmo momento, o roteador será sufocado e talvez perca pacotes. Eventualmente, o transmissor perceberá isso e enviará de novo os pacotes, mas a qualidade do serviço será instável e inadequada para áudio ou vídeo, a menos que a rede não esteja muito carregada. é desnecessário dizer que o fornecendo de qualidade de áudio adequada é algo com que as empresas de telefonia se preocupam muito, daí sua preferência por conexões.

A segunda razão para as empresas de telefonia apreciarem o serviço orientado a conexões é o fato de terem se acostumado a cobrar pelo tempo de conexão. Ao fazer uma ligação interurbana demorada (ou mesmo uma ligação local fora da América do Norte), você é cobrado por minuto. Quando as redes surgiram, elas gravitavam automaticamente em direção a um modelo no qual era fácil fazer a cobrança por minuto. Se você estabelecer uma conexão antes de enviar os dados,

nesse momento terá início o tempo de cobrança. Se não houver conexão, não será possível nenhuma cobrança.

Ironicamente, é muito dispendioso manter registros de cobrança. Se uma empresa de telefonia adotasse uma tarifa mensal fixa sem limite de ligações e sem faturamento ou manutenção de registros, ela provavelmente economizaria muito dinheiro, apesar do aumento do número de chamadas que essa política geraria. Porém, fatores políticos, reguladores e outros pesam contra isso. É interessante observar que o serviço de tarifa fixa existe em outros setores. Por exemplo, a TV a cabo é cobrada a uma taxa fixa mensal, independente de quantos programas você vê. Ela poderia ter sido projetada tendo o pay-per-view como conceito básico, mas isso não ocorreu, em parte devido às despesas de cobrança (e, considerando-se a má qualidade da maior parte dos programas de televisão, o constrangimento da operadora também não pode ser totalmente descartado). Além disso, muitos parques temáticos cobram uma tarifa diária de admissão por utilização ilimitada de suas dependências, em contraste com os parques de diversões, que cobram por cada brinquedo.

Dito isto, não deve surpreender que todas as redes projetadas pela indústria de telefonia tivessem sub-redes orientadas a conexões. O que talvez seja surpreendente é o fato da Internet também estar se movimentando nessa direção, a fim de fornecer melhor qualidade de serviço para áudio e vídeo, um assunto que voltaremos a examinar no Capítulo 5. Contudo, agora vamos estudar algumas redes orientadas a conexões.

[T4] X.25 e Frame Relay

Nosso primeiro exemplo de rede orientada a conexões é a **X.25**, a primeira rede pública de dados. Ela foi desenvolvida na década de 1970, em uma época na qual o serviço de telefonia era um monopólio em todos os lugares, e a empresa de

telefonia em cada país esperava que houvesse uma única rede de dados por país — a dela. Para usar a X.25, primeiro um computador estabelecia uma conexão com o computador remoto, isto é, fazia uma chamada telefônica. Essa conexão recebia um número de conexão que seria usado em pacotes de transferência de dados (porque várias conexões poderiam estar abertas ao mesmo tempo). Os pacotes de dados eram muito simples, consistindo em um cabeçalho de 3 bytes e até 128 bytes de dados. O cabeçalho tinha um número de conexão de 12 bits, um número de seqüência de pacote, um número de confirmação e alguns bits variados. As redes X.25 operaram por cerca de uma década com relativo sucesso.
Na década de 1980, as redes X.25 foram substituídas em grande parte por um novo tipo de rede chamado **frame relay.** A essência do frame relay é o fato de ser uma rede orientada a conexões sem controle de erros e nenhum controle de fluxo. Por se tratar de uma rede orientada a conexões, os pacotes eram entregues em ordem (quando eram entregues). As propriedades de entrega em ordem, nenhum controle de erros e nenhum controle de fluxo tornavam o frame relay semelhante a uma LAN de área extensa. Sua aplicação mais importante é a interconexão de LANs instaladas em vários escritórios de uma empresa. O frame relay desfrutou de um modesto sucesso, e ainda hoje é utilizado em alguns lugares.

[T4] ATM (Asynchronous Transfer Mode)

Outra rede orientada a conexões, e muito mais importante, é o **ATM (Asynchronous Transfer Mode).** O nome um pouco estranho é explicado pelo fato de, no sistema de telefonia, a maioria das transmissões ser síncrona (vinculada a um relógio que mantém o sincronismo) mas não o ATM.
O ATM foi projetado no início da década de 1990 e lançado em meio a uma agitação verdadeiramente incrível (Ginsburg, 1996; Goralski, 1995; Ibe, 1997;

Kim *et al.*, 1994; e Stallings, 2000). O ATM prometia resolver todos os problemas de redes e telecomunicações do mundo, mesclando voz, dados, televisão a cabo, telex, telégrafo, pombo-correio, latas conectadas por barbantes, tambores, sinais de fumaça e todos os outros meios de comunicação em um único sistema integrado que poderia fazer tudo para todos. Isso não aconteceu. Em grande parte, os problemas eram semelhantes aos que descrevemos antes em relação ao OSI, isto é, momento, tecnologia, implementação e política ruins. Tendo simplesmente derrubado as empresas de telefonia no primeiro round, muitas pessoas na comunidade da Internet viram no ATM a seqüência da luta da Internet contra as empresas de telecomunicações. Porém, isso não era verdade e, desse momento em diante, até mesmo os fanáticos por datagramas concluíram que a qualidade de serviço da Internet deixava muito a desejar. Para encurtar a história, o ATM teve muito mais sucesso que o OSI, e agora é amplamente utilizado dentro do sistema de telefonia, com freqüência para mover pacotes IP. Por ser utilizado principalmente pelas operadoras para transporte interno, muitas vezes os usuários não percebem sua existência mas, sem dúvida, ele está vivo e muito bem.

[T4] Circuitos virtuais do ATM

Tendo em vista que as redes ATM são orientadas a conexões, o envio de dados exige primeiro o envio de um pacote para configurar a conexão. À medida que o pacote de configuração passa pela sub-rede, todos os roteadores no caminho inserem uma entrada em suas tabelas internas registrando a existência da conexões e reservando os recursos necessários para ela. Com freqüência, as conexões são chamadas **circuitos virtuais**, em uma analogia com os circuitos físicos utilizados no sistema de telefonia. A maioria das redes ATM também admite **circuitos virtuais permanentes**, que são conexões permanentes entre dos

hosts (distantes). Eles são semelhantes a linhas dedicadas no universo da telefonia. Cada conexão, temporária ou permanente, tem um identificador de conexão exclusivo. Um circuito virtual é ilustrado na Figura 1.30.

[arte: ver original p. 62]

[Dísticos]

[1]Processo transmissor

[2]Host transmissor

[3]Roteador

[4]Sub-rede

[5]Host receptor

[6]Processo receptor

[7]Circuito virtual

[F]Figura 1.30

[FL]Um circuito virtual

Uma vez estabelecida uma conexão, um ou outro lado pode iniciar a transmissão de dados. A idéia básica por trás do ATM é transmitir todas as informações em pequenos pacotes de tamanho fixo chamados **células**. As células têm 53 bytes, dos quais 5 bytes formam o cabeçalho e 48 bytes são a carga útil, como mostra a Figura 1.31. Uma parte do cabeçalho é o identificador da conexão, e assim os hosts transmissor e receptor e todos os roteadores intermediários podem saber quais células pertencem a cada conexão. Essa informação permite que cada roteador saiba como rotear cada célula de entrada. O roteamento de células é feito em hardware, em alta velocidade. De fato, o principal argumento para se ter células de tamanho fixo é a facilidade para construir roteadores de hardware capazes de tratar células curtas de comprimento fixo. Pacotes IP de comprimento variável têm de ser roteados por software, um processo mais lento. Outra

vantagem do ATM é a possibilidade de configurar o hardware para copiar uma célula de entrada em várias linhas de saída, uma propriedade necessária para manipular um programa de televisão que esteja sendo transmitido por difusão a muitos receptores. Por fim, células pequenas não bloqueiam nenhuma linha por muito tempo, o que torna mais fácil garantir a qualidade de serviço.

Todas as células seguem a mesma rota até o destino. A entrega de células não é garantida, mas sua ordem sim. Se as células 1 e 2 forem enviadas nessa ordem, se ambas chegarem, elas chegarão nessa ordem, nunca 2 antes de 1. Porém, uma delas ou ambas podem se perder no caminho. Cabe aos níveis de protocolos mais altos recuperar células perdidas. Observe que, embora essa garantia não seja perfeita, é melhor que a garantia oferecida pela Internet. Lá, os pacotes não só podem se perder, mas também podem ser entregues fora de ordem. Em contraste, o ATM garante nunca entregar células fora de ordem.

[arte: ver original p. 63]

[Dísticos]

[1]Bytes

[2]5 48

[3]Cabeçalho Dados do usuário

[F]Figura 1.31

[FL]Uma célula ATM

As redes ATM são organizadas como WANs tradicionais, com linhas e switches (roteadores). As velocidades mais comuns para redes ATM são 155 Mbps e 622 Mbps, embora também seja admitidas velocidades mais altas. A velocidade de 155 Mbps foi escolhida por possibilitar a transmissão de imagens de televisão de alta definição. A escolha de 155,52 Mbps se deu por uma questão de compatibilidade com o sistema de transmissão SONET da AT&T, algo que

estudaremos no Capítulo 2. A velocidade de 622 Mbps foi escolhida para permitir que quatro canais de 155 Mbps pudessem ser enviados através dela.

[T4] O modelo de referência ATM

O ATM tem seu próprio modelo de referência, diferente do modelo OSI e também do modelo TCP/IP. Esse modelo é mostrado na Figura 1.32. Ele consiste em três camadas, a camada física, a camada ATM e a camada de adaptação ATM, além do que os usuários desejarem colocar sobre elas.

A **camada física** trata do meio físico: voltagens, sincronização de bits e uma série de outras questões. O ATM não prescreve um determinado conjunto de regras mas, em vez disso, afirma que as células ATM podem ser enviadas sozinhas por meio de um fio de cobre ou de fibra óptica, mas também podem ser reunidas na carga útil de outros sistemas de operadoras. Em outras palavras, o ATM foi projetado para ser independente do meio de transmissão.

A **camada ATM** lida com células e com o transporte de células. Ela define o layout de uma célula e revela o significado dos campos do cabeçalho. Ela também lida com o estabelecimento e a liberação de circuitos virtuais. O controle de congestionamento também está localizado nessa camada.

[arte: ver original p. 64]

[Dísticos]

[1]Gerenciamento de plano

[2]Gerenciamento de camada

[3]Plano de controle   Plano do usuário

[4]Camadas superiores   Camadas superiores

[5]CS

Camada de adaptação ATM

SAR

[6]Camada ATM

[7]TC

Camada física

PMD

[8]Subcamada

Subcamada

Subcamada

Subcamada

[9]CS: Subcamada de convergência

SAR: Subcamada de segmentação e remontagem

TC: Subcamada de convergência de transmissão

PMD: Subcamada dependente do meio físico

[F]Figura 1.32

[FL]O modelo de referência ATM

Como em geral a maioria das aplicações não trabalha diretamente com células (embora algumas possam fazê-lo), foi definida uma camada acima da camada ATM cuja finalidade é permitir aos usuários enviarem pacotes maiores que uma célula. A interface ATM segmenta esses pacotes, transmite as células individualmente e as remonta na outra extremidade. Essa é a camada **AAL (ATM Adaptation Layer).**

Ao contrário dos modelos de referência bidimensionais anteriores, o modelo ATM é, como mostra a Figura 1.32, um modelo tridimensional. O **plano do usuário** trata do transporte de dados, do fluxo de controle, da correção de erros e de outras funções do usuário. Por outro lado, o **plano de controle** se relaciona ao gerenciamento da conexões. As funções de gerenciamenro de camadas e de planos estão relacionadas ao gerenciamento de recursos e à coordenação entre as

camadas.

A camada física e a camada AAL são divididas em duas subcamadas: a camada inferior, que executa o trabalho, e uma subcamada de convergência sobre ela que fornece a interface apropriada para a camada que se encontra acima dela. As funções das camadas e subcamadas são descritas na Figura 1.33.

A subcamada **PMD (Physical Medium Dependent)** estabelece uma interface com o cabo propriamente dito. Ela ativa e desativa os bits e controla sua sincronização. Essa camada será diferente para concessionárias de comunicações e tipos de cabos específicos.

A outra subcamada da camada física é a subcamada **TC (Transmission Convergence)**. Quando as células são transmitidas, a camada TC as envia como uma seqüência de bits para a camada PMD. É fácil fazer isso. Na outra extremidade, a subcamada TC recebe um fluxo de bits puro da subcamada PMD. Sua tarefa é converter esse fluxo de bits em um fluxo de células para a camada ATM. Ela controla todas as questões relacionadas à identificação do início e do final de cada célula no fluxo de bits. No modelo ATM, essa funcionalidade se encontra na camada física. No modelo OSI e em quase todas as outras redes, a tarefa de enquadramento, ou seja, a transformação de um fluxo de bits brutos em uma seqüência de quadros ou células, cabe à camada de enlace de dados.

Como mencionamos, a camada ATM gerencia as células, inclusive sua geração e transporte. Nessa camada se encontram os aspectos mais interessantes do ATM. Ela é uma mistura das camadas de rede e de enlace de dados do OSI, mas não é dividida em subcamadas.

[arte: ver original p. 65]

[T]Tabela

Camada OSI	Camada ATM	Subcamada ATM	Funcionalidade
		CS	Fornecer a interface padrão

(convergência)

3/4 AAL

SAR Segmentação e remontagem

Controle de fluxo

2/3 ATM Geração/extração de cabeçalho de célula

Gerenciamento de caminho/circuito virtual

Multiplexação/demultiplexação de célula

Desacoplamento de taxa de células

Geração e verificação de soma de verificação de cabeçalho

2 TC Geração de células

Compactação/descompactação de células do envelope delimitador

Geração de quadro

Física

1 PMD Sincronização de bits

Acesso à rede física

[F]Figura 1.33

[FL]As camadas e subcamadas do ATM, com suas respectivas funções

A camada AAL tem as subcamadas **SAR (Segmentation And Reassembly)** e **CS (Convergence Sublayer).** A subcamada inferior divide os pacotes em células na origem e as reúne no destino. A subcamada superior permite que os sistemas ATM ofereçam diferentes tipos de serviço a diferentes aplicações (por exemplo, a

transferência de arquivos e o vídeo sob demanda lidam de maneira diferente com o tratamento de erros, a sincronização etc.).

Como o ATM deverá estar seguindo uma trajetória descendente de agora em diante, não discutiremos mais detalhes sobre ele neste livro. Apesar disso, como tem uma base instalada significativa, provavelmente ele ainda estará em uso por mais alguns anos. Para obter mais informações sobre o ATM, consulte (Dobrowski e Grise, 2001; e Gadecki e Heckart, 1997).

[T3] 1.5.3 Ethernet

Tanto a Internet quanto o ATM foram criados para redes geograficamente distribuídas. No entanto, muitas empresas, universidades e outras organizações têm grandes números de computadores que devem estar conectados. Essa necessidade deu origem à rede local. Nesta seção, faremos um breve estudo da LAN mais popular, a Ethernet. A história começa no primitivo Havaí, no início da década de 1970. Nesse caso, "primitivo" pode ser interpretado como "não tendo um sistema de telefonia funcional". Embora não ser interrompido pelo telefone o dia inteiro torne a vida mais agradável para visitantes em férias, a vida não era mais agradável para o pesquisador Norman Abramson e seus colegas da University of Hawaii, que estavam tentando conectar usuários situados em ilhas de remotas ao computador principal em Honolulu. Estender seus próprios cabos sob o Oceano Pacífico não era viável, e assim eles procuraram uma solução diferente.

A única solução que eles encontraram foi o rádio de ondas curtas. Cada terminal do usuário estava equipado com um pequeno rádio que tinha duas freqüências: ascendente (até o computador central) e descendente (a partir do computador central). Quando o usuário queria entrar em contato com o computador, ele transmitia um pacote contendo os dados no canal ascendente. Se ninguém mais

estivesse transmitindo naquele momento, o pacote provavelmente chegava e era confirmado no canal descendente. Se houvesse disputa pelo canal ascendente, o terminal perceberia a falta de confirmação e tentaria de novo. Tendo em vista que só havia um transmissor no canal descendente (o computador central), nunca ocorriam colisões nesse canal. Esse sistema, chamado ALOHANET, funcionava bastante bem sob condições de baixo tráfego, mas fica fortemente congestionado quando o tráfego ascendente era pesado.

Quase na mesma época, um estudante chamado Bob Metcalfe obteve seu título de bacharel no M.I.T. e em seguida conseguiu o título de Ph.D. em Harvard. Durante seus estudos, ele conheceu o trabalho de Abramson e ficou tão interessado que, depois de se graduar em Harvard, decidiu passar o verão no Havaí trabalhando com Abramson, antes de iniciar seu trabalho no PARC (Palo Alto Research Center) da Xerox. Ao chegar ao PARC, Metcalfe observou que os pesquisadores haviam projetado e montado o que mais tarde seria chamado computador pessoal. No entanto, as máquinas estavam isoladas. Usando seu conhecimento do trabalho realizado por Abramson, ele e seu colega David Boggs, projetaram e implementaram a primeira rede local (Metcalfe e Boggs, 1976).

O sistema foi chamado **Ethernet**, uma menção ao *éter luminoso*, através do qual os antigos diziam que a radiação eletromagnética se propagava. (Quando o físico britânico do Século XIX James Clerk Maxwell descobriu que a radiação eletromagnética podia ser descrita por um equação de onda, os cientistas imaginaram que o espaço teria de ser preenchido com algum meio etéreo no qual a radiação se propagaria. Só depois da famosa experiência de Michelson-Morley em 1887, os físicos descobriram que a radiação eletromagnética era capaz de se propagar no vácuo.) Nesse caso, o meio de transmissão não era o vácuo, mas um cabo coaxial grosso (o éter) com até 2,5 km de comprimento (com repetidores a cada 500 metros). Até 256 máquinas podiam ser conectadas ao sistema por meio

de transceptores presos ao cabo. Um cabo com várias máquinas conectadas a ele em paralelo é chamado **cabo multiponto.** O sistema funcionava a 2,94 Mbps. A Figura 1.34 mostra um esboço de sua arquitetura. A Ethernet tinha um aperfeiçoamento importante em relação à ALOHANET: antes de transmitir, primeiro um computador inspecionava o cabo para ver se alguém mais já estava transmitindo. Nesse caso, o computador ficava impedido até a transmissão atual terminar. Isso evitava interferências com transmissões em andamento, o que proporcionava uma eficiência muito maior. A ALOHANET não funcionava assim porque era impossível para um terminal em uma ilha detectar a transmissão de um terminal em outra ilha distante. Com um único cabo, esse problema não existe.

[arte: ver original p. 67]

[Dísticos]

[1]Transceptor

[2]Cabo de interface

[3]Éter

[F]Figura 1.34

[FL] Arquitetura da Ethernet original

Apesar da escuta do computador antes de transmitir, ainda surge um problema: o que acontece se dois ou mais computadores esperarem até a transmissão atual se completar e depois todos começarem as transmitir ao mesmo tempo? A solução é fazer cada computador se manter na escuta durante sua própria transmissão e, se detectar interferência, bloquear o éter para alertar todos os transmissores. Em seguida, recuar e esperar um tempo aleatório antes de tentar novamente. Se ocorrer uma segunda colisão, o tempo aleatório de espera será duplicado e assim por diante, até separar as transmissões concorrentes e dar a uma delas a chance

de iniciar sua transmissão.

A Ethernet da Xerox foi tão bem-sucedida que a DEC, a Intel e a Xerox criaram em 1978 um padrão para uma Ethernet de 10 Mbps, chamado **padrão DIX.** Com duas pequenas alterações, o padrão DIX se tornou o padrão IEEE 802.3 em 1983. Infelizmente para a Xerox, ela já tinha um histórico de criar invenções originais (como o computador pessoal) e depois deixar de comercializá-las, uma história contada em *Fumbiling the Future* (Smith e Alexander, 1988). Quando a Xerox mostrou pouco interesse em fazer algo com a Ethernet além de ajudar a padronizá-la, Metcalfe formou sua própria empresa, a 3Com, para vender adaptadores Ethernet destinados a PCs. A empresa vendeu mais de 100 milhões desses adaptadores.

A Ethernet continuou a se desenvolver e ainda está em desenvolvimento. Surgiram novas versões a 100 Mbps, 1000 Mbps e a velocidades ainda mais altas. O cabeamento também melhorou, e foram acrescentados recursos de comutação e outras características. Descreveremos a Ethernet em detalhes no Capítulo 4.

Vale a pena mencionar de passagem que a Ethernet (IEEE 802.3) não é o único padrão de LAN. O comitê também padronizou um barramento de símbolos (802.4) e um anel de símbolos (802.5). A necessidade de três padrões mais ou menos incompatíveis tem pouca relação com a tecnologia e muita com a política. Na época da padronização, a General Motors estava desenvolvendo uma LAN na qual a topologia era a mesma da Ethernet (um cabo linear), mas os computadores transmitiam por turnos, pela passagem de um pequeno pacote chamado **símbolo** ou **token** (ficha) de um computador para outro. Um computador só podia transmitir se tivesse a posse do símbolo, e isso evitava colisões. A General Motors anunciou que esse esquema era essencial para a fabricação de automóveis e não estava preparada para desistir dessa posição. Apesar desse anúncio, o 802.4 basicamente desapareceu.

De modo semelhante, a IBM também tinha sua preferência: sua rede token ring (anel de símbolos) patenteada. O símbolo era repassado pelo anel e qualquer computador que tivesse o símbolo tinha permissão para transmitir antes de colocar o símbolo de volta no anel. Diferente do 802.4, esse esquema, padronizado como 802.5, está é usado em algumas instalações da IBM, mas não é encontrado em praticamente nenhum outro lugar além da IBM. Porém, há um trabalho em andamento para produzir uma versão de gigabit (802.5v), mas parece improvável que ela consiga competir com a Ethernet. Em suma, houve uma guerra entre Ethernet, token bus (barramento de símbolos) e token ring (anel de símbolos), e a Ethernet venceu, principalmente porque surgiu primeiro e seus adversários não eram tão bons quanto ela.

[T3]1.5.4 LANs sem fios: 802.11

Quase na mesma época em que surgiram os notebooks, muitas pessoas sonhavam com o dia em que entrariam em um escritório e magicamente seu notebook se conectaria à Internet. Em conseqüência disso, diversos grupos começaram a trabalhar para descobrir maneiras de alcançar esse objetivo. A abordagem mais prática é equipar o escritório e os notebooks com transmissores e receptores de rádio de ondas curtas para permitir a comunicação entre eles. Esse trabalho levou rapidamente à comercialização de LANs sem fios por várias empresas.

O problema era encontrar duas delas que fossem compatíveis. Essa proliferação de padrões significava que um computador equipado com um rádio da marca *X* não funcionaria em uma sala equipada com uma estação base da marca *Y*. Finalmente, a indústria decidiu que um padrão de LAN sem fio poderia ser uma boa idéia, e assim o comitê do IEEE que padronizou as LANs sem fios recebeu a tarefa de elaborar um padrão de LANs sem fios. O padrão recebeu o nome

802.11. Um apelido comum para ele é **WiFi**. Trata-se de um padrão importante e que merece respeito, e assim vamos chamá-lo por seu nome correto, 802.11.

O padrão proposto tinha de funcionar em dois modos:

1. Na presença de uma estação base.

2. Na ausência de uma estação base.

No primeiro caso, toda a comunicação deveria passar pela estação base, chamada **ponto de acesso** na terminologia do 802.11. No outro caso, os computadores simplesmente transmitiriam diretamente uns para os outros. Agora, esse modo costuma ser chamado **interligação de redes *ad hoc***. Um exemplo típico é de duas ou mais pessoas juntas em uma sala não equipada com uma LAN sem fio, fazendo seus computadores se comunicarem diretamente. Os dois modos estão ilustrados na Figura 1.35.

A primeira decisão foi a mais fácil: como denominá-lo. Todos os outros padrões de LANs tinham números como 802.1, 802.2, 802.3, até 802.10; assim, o padrão de LAN sem fio recebeu a denominação 802.11. O resto era mais difícil.

Em particular, alguns dos muitos desafios que tinham de ser enfrentados eram: descobrir uma banda de freqüências adequada que estivesse disponível, de preferência em todo o mundo; lidar com o fato de que os sinais de rádio têm um alcance finito; assegurar que a privacidade dos usuários seria mantida, levando em conta a duração limitada da bateria; preocupação com a segurança humana (as ondas de rádio causam câncer?); compreender as implicações da mobilidade dos computadores; por fim, construir um sistema com largura de banda suficiente para ser economicamente viável.

[arte: ver original p. 69]

[Dísticos]

[1]Estação base Para a rede fisicamente conectada

[2](a) (b)

[F]Figura 1.35

[FL](a) Rede sem fio com uma estação base. (b) Rede *ad hoc*

Na época em que o processo de padronização começou (meados da década de 1990), a Ethernet já havia dominado o mercado de redes locais, e assim o comitê decidiu tornar o 802.11 compatível com a Ethernet acima da camada de link de dados. Em particular, deve ser possível enviar um pacote IP pela LAN sem fio, do mesmo modo que um computador conectado envia um pacote IP pela Ethernet. Apesar disso, existem várias diferenças inerentes em relação à Ethernet na camada física e na camada de enlace de dados, e essas diferenças tinham de ser tratadas pelo padrão.

Em primeiro lugar, um computador na Ethernet sempre escuta o éter antes de transmitir. Somente se o éter estiver ocioso o computador inicia a transmissão. No caso de LANs sem fios, essa idéia não funciona muito bem. Para ver por que, examine a Figura 1.36. Suponha que o computador *A* esteja transmitindo para o computador *B*, mas que o alcance do rádio do transmissor de *A* seja curto demais para alcançar o computador *C*. Se *C* quiser transmitir para *B*, ele poderá escutar o éter antes de começar, mas o fato de não ouvir nada não significa que sua transmissão será bem-sucedida. O padrão 802.11 tinha de resolver esse problema.

O segundo problema que tinha de ser resolvido era a possibilidade de objetos sólidos refletirem o sinal de rádio, de forma que o sinal pudesse ser recebido várias vezes (ao longo de diversos caminhos). Essa interferência resulta naquilo que se denomina **atenuação multiponto**.

O terceiro problema é que grande parte do software não está ciente da mobilidade. Por exemplo, muitos processadores de textos têm uma lista de impressoras que os usuários podem escolher para imprimir um arquivo. Quando

o computador no qual o processador de textos funciona é levado para um novo ambiente, a lista interna de impressoras se torna inválida.

[arte: ver original p. 70a]

[Dísticos]

[1]Alcance do rádio de A

[2]Alcance do rádio de C

[3]A B C

[F]Figura 1.36

[FL]O alcance de um único rádio pode não cobrir o sistema inteiro

O quarto problema é que, se um notebook for afastado da estação base que ele está utilizando e entrar na faixa de alcance de uma estação base diferente, será necessária alguma forma de @@@transferência (handoff). Embora ocorra com telefones celulares, esse problema não acontece com a Ethernet e precisa ser resolvido. Em particular, a rede prevista consiste em várias células, cada uma com sua própria estação base, mas com as estações base conectadas pela Ethernet, como mostra a Figura 1.37. Visto do exterior, o sistema inteiro deve ser semelhante a uma única Ethernet. A conexão entre o sistema 802.11 e o mundo exterior é chamada **portal**.

[arte: ver original p. 70b]

[Dísticos]

[1]Ethernet

[2]Estação base

[3]Portal

[4]Célula

[F]Figura 1.37

[FL]Uma rede 802.11 de várias células

Após algum trabalho, o comitê apresentou um padrão em 1997 que tratou dessas e de outras questões. A LAN sem fio que ele descreveu funcionava a 1 Mbps ou 2 Mbps. Quase imediatamente, as pessoas reclamaram que ela era muito lenta, e assim começou o trabalho em padrões mais rápidos. Uma divisão se desenvolveu dentro do comitê, resultando em dois novos padrões publicados em 1999. O padrão 802.11a utiliza uma faixa de freqüências mais larga e funcionava em velocidades de 54 Mbps. O padrão 802.11b utiliza a mesma faixa de freqüências que o 802.11, mas emprega uma técnica de modulação diferente para alcançar 11 Mbps. Algumas pessoas vêem esse fato como algo psicologicamente importante, tendo em vista que 11 Mbps é mais rápido que a Ethernet fisicamente conectada original. É provável que o padrão original de 1 Mbps, o 802.11, desapareça rapidamente, mas ainda não está claro qual dos novos padrões ocupará seu lugar.

Para tornar a questão ainda mais complicada do que já era, o comitê 802 apresentou ainda outra variante, o 802.11g, que utiliza a técnica de modulação do 802.11a, mas emprega a faixa de freqüências do 802.11b. Voltaremos a estudar o 802.11 em detalhes no Capítulo 4.

Não há mais dúvida que o 802.11 causará uma revolução na computação e no acesso à Internet. Aeroportos, estações de trem, hotéis, centros comerciais e universidades estão instalando rapidamente essas redes. Até mesmo as lojas de cibercafés estão instalando o 802.11, para que os yuppies reunidos possam navegar pela Web enquanto apreciam sua bebida. É provável que o 802.11 faça pela Internet o que os notebooks fizeram pela computação: torná-la móvel.

[T2] 1.6 Padronização de redes

Existem muitos fabricantes e fornecedores de redes, cada qual com sua própria

concepção de como tudo deve ser feito. Sem coordenação, haveria um caos completo, e os usuários nada conseguiriam. A única alternativa de que a indústria dispõe é a criação de alguns padrões de rede.

Além de permitirem que diferentes computadores se comuniquem, os padrões também ampliam o mercado para os produtos que aderem a suas regras. Um mercado mais amplo estimula a produção em massa, proporciona uma economia no processo de produção e permite a criação de implementações VLSI e de outros benefícios que diminuem o preço e aumentam mais ainda a aceitação de um produto. Nas próximas seções, faremos uma rápida análise sobre o mundo da padronização internacional que, apesar de pouco conhecido, é de grande importância.

Os padrões se dividem em duas categorias: *de facto* e *de jure*. Os padrões **de facto** (que corresponde a "de fato" em português) são aqueles que se consagraram naturalmente, sem nenhum plano formal. O IBM PC e seus sucessores são os padrões *de facto* para computadores pessoais em pequenos escritórios e nos lares, pois dezenas de fabricantes resolveram copiar as máquinas da IBM praticamente na íntegra. O UNIX é o padrão *de facto* dos sistemas operacionais utilizados nos departamentos de ciência da computação das universidades.

Os padrões **de jure** (que correspondem a "por lei" em português), ao contrário, são padrões legais e formais adotados por uma instituição de padronização autorizada. Em geral, as autoridades de padronização internacional são divididas em duas classes: as que foram estabelecidas por tratados entre governos nacionais e as organizações voluntárias, que foram criadas independentemente de tratados. Na área de padrões de redes de computadores, há diversas organizações de ambos os tipos, que serão discutidas a seguir.

[T3] 1.6.1 Quem é quem no mundo das telecomunicações

O status legal das empresas telefônicas do mundo varia consideravelmente de um país para outro. De um lado, estão os Estados Unidos, que têm 1.500 empresas telefônicas separadas. Antes de ser desmembrada em 1984, a AT&T, que na época era a maior empresa do mundo, tinha o domínio absoluto do mercado. Ela fornecia serviços de telefonia para cerca de 80% dos Estados Unidos e se estendia por metade de sua área geográfica, com todas as outras empresas combinadas atendendo aos clientes restantes (em sua maioria, das regiões rurais). Desde o desmembramento, a AT&T continua a oferecer serviços de longa distância, embora atualmente ela tenha de disputar esse mercado com outras empresas. As sete empresas de telefonia locais criadas a partir da divisão da AT&T e numerosas empresas independentes fornecem serviços na área de telefonia local e celular. Devido a freqüentes fusões e outras mudanças, a indústria se encontra em um constante estado de fluxo.

As empresas americanas que oferecem serviços de comunicação ao público são chamadas **concessionárias de comunicações**. Os serviços que oferecem e os preços que praticam são descritos por um documento chamado **tarifa**, que deve ser aprovado pela Federal Communications Commission para o tráfego interestadual e internacional, e pelas comissões estaduais de serviços de utilidade pública para o tráfego dentro de cada um dos estados americanos.

No outro extremo, estão os países em que o governo federal detém o monopólio de toda a área de comunicações, incluindo correio, telégrafo, telefone e muitas vezes rádio e televisão. A maior parte do mundo se enquadra nessa categoria. Em alguns casos, as telecomunicações são comandadas por uma empresa nacionalizada, mas em outros elas são controladas por uma estatal, em geral conhecida como **PTT (Post, Telegraph & Telephone)**. No mundo inteiro, a tendência é de liberalização e competição, encerrando o monopólio do governo.

A maioria dos países europeus tem agora suas PTTs (parcialmente) privatizadas, mas em outros lugares o processo ainda está ganhando impulso lentamente.

Com todos esses fornecedores de serviços, é cada vez maior a necessidade de oferecer compatibilidade em escala mundial para garantir que pessoas (e computadores) de diferentes países possam se comunicar. Na verdade, essa necessidade já existe há muito tempo. Em 1865, representantes de diversos governos europeus se reuniram para formar a predecessora da atual **ITU (International Telecommunication Union)**. A missão da ITU era padronizar as telecomunicações internacionais, até então dominadas pelo telégrafo. Já naquela época estava claro que, se metade dos países utilizasse o código Morse e a outra metade usasse algum outro código, haveria problemas de comunicação. Quando o telefone passou a ser um serviço internacional, a ITU também se encarregou de padronizar a telefonia. Em 1947, a ITU tornou-se um órgão das Nações Unidas.

A ITU tem três setores principais:

1. Setor de radiocomunicações (ITU-R).
2. Setor de padronização de telecomunicações (ITU-T).
3. Setor de desenvolvimento (ITU-D).

A ITU-R regula a alocação de freqüências de rádio em todo o mundo entre grupos de interesses conflitantes. Vamos nos concentrar principalmente na ITU-T, que controla os sistemas de telefonia e de comunicação de dados. De 1956 a 1993, a ITU-T foi conhecida como CCITT, acrônimo de Comité Consultatif International Télégraphique et Téléphonique, seu nome em francês. No dia 1º de março de 1993, o CCITT foi reorganizado para se tornar menos burocrático e teve seu nome alterado para refletir as novas funções. O ITU-T e o CCITT publicavam recomendações na área de telefonia e de comunicações de dados. Ainda hoje, nos deparamos com uma série de recomendações do CCITT, como o CCITT X.25, muito embora desde 1993 as recomendações ostentem o rótulo ITU-T.

A ITU-T tem quatro classes de membros:

1. Governos nacionais.
2. Membros setoriais.
3. Membros associados.
4. Agências reguladoras.

A ITU-T tem cerca de 200 membros governamentais, incluindo quase todos os membros das Nações Unidas. Tendo em vista que os Estados Unidos não têm uma PTT, outro grupo teve de representá-lo na ITU-T. Essa tarefa coube ao Departamento de Estado, provavelmente porque a ITU-T se relacionava com países estrangeiros, a especialidade do Departamento de Estado. Existem aproximadamente 500 membros setoriais, incluindo as empresas de telefonia (por exemplo, AT&T, Vodafone, WorldCom), fabricantes de equipamentos de telecomunicações (por exemplo, Cisco, Nokia, Nortel), fornecedores de computadores (por exemplo, Compaq, Sun, Toshiba), fabricantes de chips (por exemplo, Intel, Motorola, TI), empresas de mídia (por exemplo, AOL Time Warner, CBS, Sony) e outras empresas interessadas (por exemplo, Boeing, Samsung, Xerox). Várias organizações científicas sem fins lucrativos e consórcios industriais também são membros setoriais (por exemplo, IFIP e IATA). Os membros associados são organizações menores, interessadas em um grupo de estudos específico. As agências reguladoras são as que definem as normas do negócio de telecomunicações, como a Federal Communications Commission nos EUA.

A tarefa da ITU-T é definir recomendações técnicas para interfaces de telefonia, telégrafo e comunicação de dados. Em geral, essas recomendações se transformam em padrões reconhecidos internacionalmente como, por exemplo, o V.24 (também conhecido como EIA RS-232 nos EUA), que especifica a posição e o significado dos diversos pinos no conector usado pela maioria dos terminais assíncronos e modems externos.

Vale lembrar que tecnicamente as recomendações da ITU-T são apenas sugestões que os governos podem adotar ou ignorar, se assim o desejarem (porque os governos são como garotos de treze anos – eles não gostam de receber ordens). Na prática, um país que deseja adotar um padrão de telefonia diferente do restante do mundo tem toda liberdade de fazê-lo, mas ficará isolado de todos os outros. Essa opção pode ser válida na Coréia do Norte, mas seria a fonte de muitos problemas em outros lugares. Na verdade, chamar os padrões criados pela ITU-T de "recomendações" não passa de um eufemismo que serve para manter a paz com as forças nacionalistas de muitos países.

O trabalho da ITU-T é feito em seus 14 grupos de estudo que, em geral, abrangem 400 pessoas. Existem no momento 14 grupos de estudo que abrangem tópicos que vão da cobrança de tarifas telefônicas até serviços de multimídia. Para tornar possível a obtenção de algum resultado, os grupos de estudo se dividem em setores de trabalho que, por sua vez, se dividem em equipes de especialistas que, por sua vez, se dividem em outros grupos. Uma vez burocracia, sempre burocracia.

Apesar de todas essas dificuldades, a ITU-T tem suas realizações. Desde seu começo, ela produziu cerca de 3000 recomendações que ocupam aproximadamente 60.000 páginas de papel. Muitas delas são amplamente utilizadas na prática. Por exemplo, o popular padrão de modems de 56 kbps V.90 é uma recomendação da ITU.

Quando a transição iniciada na década de 1980 for concluída e as telecomunicações deixarem de ser uma questão interna de cada país para ganhar o status de uma questão global, os padrões ganharão cada vez mais importância, e um número cada vez maior de organizações tenderá a participar do processo de definição de padrões. Para obter mais informações sobre a ITU, consulte (Irmer, 1994).

[T3] 1.6.2 Quem é quem no mundo dos padrões internacionais

Os padrões internacionais são produzidos e publicados pela **ISO (International Standards Organization*)**, uma organização voluntária independente, fundada em 1946. Seus membros são as organizações nacionais de padrões dos 89 países membros. Dentre eles estão as seguintes organizações: ANSI (EUA), BSI (Grã-Bretanha), AFNOR (França), DIN (Alemanha) e mais 85 participantes.

* [Nota de rodapé]

Para os puristas, o verdadeiro nome é International Organization for Standardization.

A ISO publica padrões sobre uma vasta gama de assuntos, que vão desde parafusos e porcas (literalmente) ao revestimento usado nos postes telefônicos [sem mencionar sementes de cacau (ISO 2451), redes de pesca (ISO 1530), roupas íntimas femininas (ISO 4416) e vários outros assuntos que ninguém imaginaria que fossem padronizados]. Ela já publicou mais de 13 mil padrões, incluindo os padrões OSI. A ISO tem quase 200 comissões técnicas, numeradas por ordem de criação — cada uma delas lida com um assunto específico. A TC1 lida com porcas e parafusos (padronizando as medidas da rosca). A TC97 trata de computadores e processamento de informações. Cada TC tem subcomissões (SCs) que, por sua vez, se dividem em grupos de trabalho (WGs).

O trabalho da ISO é feito nos grupos de trabalho, em torno dos quais se reúnem 100 mil voluntários de todo o mundo. Muitos desses "voluntários" foram escalados para trabalhar em questões da ISO pelos seus empregadores, cujos produtos estão sendo padronizados. Outros são funcionários públicos ansiosos por descobrir um meio de transformar em padrão internacional o que é feito em seus países de origem. Especialistas acadêmicos também têm participação ativa

em muitos grupos de trabalho.

Nas questões relacionadas aos padrões de telecomunicações, a ISO e a ITU-T costumam trabalhar em conjunto (a ISO é membro da ITU-T), para evitar a ironia de dois padrões internacionais oficiais serem mutuamente incompatíveis.

O representante dos Estados Unidos na ISO é o **ANSI (American National Standards Institute)** que, apesar do nome, é uma organização não governamental sem fins lucrativos. Seus membros são fabricantes, concessionárias de comunicações e outras partes interessadas. Os padrões ANSI freqüentemente são adotados pela ISO como padrões internacionais.

O procedimento usado pela ISO para a adoção de padrões foi criado de modo a obter o maior consenso possível. O processo começa quando uma das organizações de padrões nacionais sente a necessidade de um padrão internacional em alguma área. Em seguida, é formado um grupo de trabalho com a finalidade de produzir um **CD (Committee Draft)**. Depois, o CD é distribuído a todas as entidades associadas, que têm seis meses para analisá-lo. Se ele for aprovado por uma ampla maioria, um documento revisado, chamado **DIS (Draft International Standard)** será produzido e distribuído para receber comentários e ser votado. Com base nos resultados, o texto final do **IS (International Standard)** é preparado, aprovado e publicado. Nas áreas de grande controvérsia, o CD ou o DIS passa por diversas revisões até obter o número de votos necessário, em um processo que pode durar anos.

O **NIST (National Institute of Standards and Technology)** é um órgão do Departamento de Comércio dos Estados Unidos. Ele já foi chamado de National Bureau of Standards e emite padrões que controlam as compras feitas pelo governo dos Estados Unidos, exceto as do Departamento de Defesa, que tem seus próprios padrões.

Outra estrela do mundo dos padrões é o **IEEE (Institute of Electrical and**

Electronies Engineers), a maior organização profissional do mundo. Além de publicar uma série de jornais e promover diversas conferências a cada ano, o IEEE tem um grupo de padronização que desenvolve padrões nas áreas de engenharia elétrica e de informática. O comitê 802 do IEEE padronizou vários tipos de LANs. Estudaremos alguns de seus resultados mais adiante neste livro. O trabalho é feito por um conjunto de grupos de trabalho, listados na Figura 1.38. A taxa de sucesso dos diversos grupos de trabalho do 802 tem sido baixa; ter um número 802.x não é garantia de sucesso. Porém, o impacto das histórias de sucesso (em especial do 802.3 e do 802.11) é enorme.

[T3] 1.6.3 Quem é quem no mundo dos padrões da Internet

A Internet mundial tem seus próprios mecanismos de padronização, que são bastante diferentes dos adotados pela ITU-T e pela ISO. Grosso modo, pode-se dizer que as pessoas que participam das reuniões de padronização da ITU ou da ISO se apresentam de paletó e gravata. Já as pessoas que participam das reuniões de padronização da Internet usam jeans (exceto quando o tempo está quente e elas utilizam bermudas e chinelos).

As reuniões da ITU-T e da ISO são freqüentadas por pessoas ligadas à iniciativa privada e ao governo, cuja especialidade é a padronização. Para essas pessoas, a padronização é algo sagrado e a ela dedicam suas vidas. Por outro lado, o pessoal da Internet tem uma natureza anárquica. No entanto, com centenas de milhões de pessoas fazendo tudo por sua conta, não é possível haver muita comunicação. Assim, às vezes é preciso criar algumas regras para facilitar o funcionamento da rede. Por essa razão, os padrões – apesar dos pesares – acabam se fazendo necessários.

Quando a ARPANET foi configurada, o Departamento de Defesa dos EUA criou uma comissão informal para supervisioná-la. Em 1983, a comissão passou a ser

chamada **IAB (Internet Activities Board)** e teve seus poderes ampliados, ou seja, foi possível manter os pesquisadores envolvidos com a ARPANET e a Internet mais ou menos voltados para uma mesma direção, uma tarefa bastante árdua, diga-se de passagem. Mais tarde, o significado do acrônimo "IAB" mudou para **Internet Architecture Board.**

[arte: ver original p. 76]

[T]Tabela

Número	**Assunto**
802.1	Avaliação e arquitetura de LANs
802.2 ↓	Controle de link lógico
802.3 *	Ethernet
802.4 ↓	Token bus (barramento de símbolos; foi usado por algum tempo em unidades industriais)
802.5	Token ring (anel de símbolos, a entrada da IBM no mundo das LANs)
802.6 ↓	Fila dual barramento dual (primeira rede metropolitana)
802.7 ↓	Grupo técnico consultivo sobre tecnologias de banda larga
802.8 †	Grupo técnico consultivo sobre tecnologias de fibra óptica
802.9 ↓	LANs isócronas (para aplicações de tempo real)
802.10 ↓	LANs virtuais e segurança
802.11 *	LANs sem fios
802.12 ↓	Prioridade de demanda (AnyLAN da Hewlett Packard)
802.13	Número relacionado à má sorte. Ninguém o quis
802.14 ↓	Modems a cabo (extinto: um consórcio industrial conseguiu chegar primeiro)
802.15 *	Redes pessoais (Bluetooth)
802.16 *	Rádio de banda larga
802.17	Anel de pacote elástico

[F]Figura 1.38

[FL]Os grupos de trabalho 802. Os grupos importantes estão marcados com *. Aqueles marcados com ↓ estão hibernando. O grupo marcado com † desistiu e se licenciou

Cada um dos cerca de dez membros do IAB coordenou uma força-tarefa sobre alguma questão importante. O IAB promovia diversas reuniões anuais para discutir os resultados e prestar contas ao Departamento de Defesa dos EUA e à NSF, que na época estavam financiando a maior parte de suas atividades. Quando havia necessidade de um padrão (por exemplo, um novo algoritmo de roteamento), os membros do IAB o elaboravam e, em seguida, anunciavam a mudança aos estudantes universitários, de modo que os envolvidos na produção do software pudessem implementá-lo. As comunicações eram feitas por uma série de relatórios técnicos, chamados **RFCs (Request For Comments)**. As RFCs são armazenados on-line, e todas as pessoas interessadas podem ter acesso a elas em *www.ietf.org/rfc*. Elas são numeradas em ordem cronológica de criação, e já estão na casa dos 3 mil. Vamos nos referir a muitas RFCs neste livro.

Por volta de 1989, a Internet havia crescido tanto que teve de abdicar de seu estilo altamente informal. Muitos fabricantes estavam oferecendo produtos TCP/IP e não queriam mudá-los só porque uma dezena de pesquisadores acreditava ter uma idéia melhor. No verão de 1989, o IAB se reorganizou mais uma vez. Os pesquisadores se reuniram em torno da **IRTF (Internet Research Task Force)**, que se transformou em uma subsidiária do IAB, assim como a **IETF (Internet Engineering Task Force)**. O IAB foi novamente ocupado por pessoas que representavam uma faixa mais ampla de organizações que a simples comunidade de pesquisa. Inicialmente, os membros do grupo teriam um mandato indireto de dois anos, sendo os novos membros indicados pelos antigos. Mais tarde, foi

criada a **Internet Society**, integrada por pessoas interessadas na Internet. De certa forma, a Internet Society pode ser comparada à ACM ou ao IEEE. Ela é administrada por conselheiros eleitos que, por sua vez, indicam os membros do IAB.

A idéia dessa divisão era fazer com que a IRTF se concentrasse em pesquisas a longo prazo, enquanto a IETF iria lidar com questões de engenharia a curto prazo. A IETF foi dividida em grupos de trabalho, cada qual deveria resolver um problema específico. Os coordenadores desses grupos de trabalho formariam uma espécie de comitê geral para orientar o esforço de engenharia. Dentre os assuntos estudados pelos grupo de trabalho estão novas aplicações, informações para o usuário, integração do OSI, roteamento e endereçamento, segurança, gerenciamento de redes e padrões. Formaram-se tantos grupos de trabalho (mais de 70) que foi necessário agrupá-los em áreas, cujos coordenadores passaram a integrar o comitê geral.

Além disso, foi adotado um processo de padronização mais formal, semelhante aos da ISO. Para se tornar um **Proposed Standard** (padrão proposto), a idéia básica deve ser completamente explicada em uma RFC e despertar na comunidade um interesse suficiente para merecer algum tipo de consideração. Para chegar ao estágio de **Draft Standard**, o padrão proposto precisa ser completamente testado por, no mínimo, dois sites independentes durante quatro meses. Se o IAB for convencido de que a idéia é viável e de que o software funciona, ele poderá atribuir à RFC em questão o status de Internet Standard (padrão da Internet). Alguns padrões da Internet foram adotados pelo Departamento de Defesa dos EUA (MIL-STD), tornando-se obrigatórios para seus fornecedores. David Clark fez o seguinte comentário, hoje famoso, sobre a padronização da Internet: "consenso rígido e código funcional".

[T2] 1.7 Unidades métricas

Para evitar qualquer confusão, vale a pena declarar explicitamente que neste livro, como na ciência da computação em geral, as unidades métricas são usadas em lugar de unidades tradicionais inglesas. Os principais prefixos métricos estão listados na Figura 1.39. Em geral, os prefixos são abreviados por sua letra inicial, com as unidades maiores que 1 em maiúsculas (KB, MB etc.). Uma exceção (por razões históricas) é a unidade kbps para indicar kilobits/s. Desse modo, uma linha de comunicação de 1 Mbps transmite 10^6 bits/s e um clock de 100 psegundos (ou 100 ps) pulsa a cada 10^{-10} segundos. Tendo em vista que os prefixos mili e micro começam ambos pela letra "m", foi preciso fazer uma escolha. Normalmente, "m" representa mili e "µ" (a letra grega mu) representa micro.

Também vale a pena assinalar que, para medir tamanhos de memória, disco, arquivos e bancos de dados, uma prática comum na indústria, as unidades têm significados um pouco diferentes. Nesses casos, kilo significa 2^{10} (1024), em vez de 10^3 (1000), porque as memórias sempre são medidas em potências de dois. Desse modo, uma memória de 1 KB contém 1024 bytes, e não 1000 bytes. De modo semelhante, uma memória de 1 MB contém 2^{20} (1.048.576) bytes, uma memória de 1 GB contém 2^{30} (1.073.741.824) bytes e um banco de dados de 1 TB contém 2^{40} (1.099.511.627.776) bytes. No entanto, uma linha de comunicação de 1 kbps transmite 1000 bits por segundo, e uma LAN de 10 Mbps funciona a 10.000.000 bits/s, porque essas velocidades não são potências de dois. Infelizmente, muitas pessoas tendem a misturar esses dois sistemas, especialmente para tamanhos de discos. Para evitar ambigüidade, usaremos neste livro os símbolos KB, MB e GB para 2^{10}, 2^{20} e 2^{30} bytes, respectivamente, e os símbolos kbps, Mbps e Gbps para 10^3, 10^6 e 10^9 bits/s, respectivamente.

[arte: ver original p. 78]

[T]Tabela

Exp.	Explícito	Prefixo
10^{-3}	0,001	mili
10^{-6}	0,000001	micro
10^{-9}	0,000000001	nano
10^{-12}	0,000000000001	pico
10^{-15}	0,000000000000001	femto
10^{-18}	0,000000000000000001	atto
10^{-21}	0,000000000000000000001	zepto
10^{-24}	0,000000000000000000000001	yocto
Exp.	**Explícito**	**Prefixo**
10^{3}	1.000	Kilo
10^{6}	1.000.000	Mega
10^{9}	1.000.000.000	Giga
10^{12}	1.000.000.000.000	Tera
10^{15}	1.000.000.000.000.000	Peta
10^{18}	1.000.000.000.000.000.000	Exa
10^{21}	1.000.000.000.000.000.000.000	Zetta
10^{24}	1.000.000.000.000.000.000.000.000	Yotta

[F]Figura 1.39

[FL] Os principais prefixos métricos

[T2] 1.8 Visão geral dos outros capítulos do livro

Este livro descreve os princípios e a prática das redes de computadores. A maioria dos capítulos começa com uma discussão dos princípios relevantes, seguida por uma série de exemplos ilustrativos. Em geral, esses exemplos são extraídos da Internet de redes sem fios, pois essas são redes importantes e muito diferentes.

Serão apresentados outros exemplos onde for relevante.

A estrutura deste livro segue o modelo híbrido da Figura 1.24. A partir do Capítulo 2, vamos examinar toda a hierarquia de protocolos, começando pela parte inferior. O segundo capítulo contém uma rápida análise do processo de comunicação de dados. Ele abrange sistemas fisicamente conectados, sem fios e de transmissão por satélite. Esse material está voltado para a camada física, apesar de examinarmos apenas sua arquitetura e deixarmos de lado os aspectos de hardware. Diversos exemplos da camada física também são discutidos, como a rede de telefonia pública comutada, os telefones celulares e a rede de televisão a cabo.

O Capítulo 3 discute a camada de enlace de dados e seus protocolos por meio de uma série de exemplos cada vez mais complexos. Também apresentamos uma análise desses protocolos. Depois, são descritos alguns protocolos reais importantes, incluindo o HDLC (usado em redes de baixa e média velocidade) e o PPP (usado na Internet).

O Capítulo 4 é dedicado à subcamada de acesso ao meio, que faz parte da camada de enlace de dados. Nele, a questão básica é como determinar quem pode usar a rede em seguida, quando a rede consiste em um único canal compartilhado, como ocorre na maioria das LANs e em algumas redes de satélites. Há muitos exemplos das áreas de LANs fisicamente conectadas, LANs sem fios (especialmente a Ethernet), MANs sem fios, Bluetooth e redes de satélites. Nesse capítulo, também descrevemos pontes e switches de enlace de dados, usados para conectar LANs.

O Capítulo 5 trata da camada de rede, em especial o roteamento, sendo abordados muitos algoritmos de roteamento, tanto estático quanto dinâmico. Porém, mesmo com bons algoritmos de roteamento, se for oferecido mais tráfego do que a rede pode manipular, será possível haver congestionamento, e assim

discutiremos o congestionamento e como evitá-lo. Melhor ainda do que apenas impedir o congestionamento é garantir uma certa qualidade de serviço. Também discutiremos esse tópico no capítulo. A conexão de redes heterogêneas para formar inter-redes leva a numerosos problemas que também serão analisados. A camada de rede na Internet recebe extenso tratamento.

O Capitulo 6 é dedicado à camada de transporte. Nele, é dada ênfase especial aos protocolos orientados a conexões, utilizados por muitas aplicações. Um exemplo de serviço de transporte e sua implementação são detalhadamente analisados. É dado o código real desse exemplo simples, a fim de mostrar como ele poderia ser implementado. Estudaremos os protocolos de transporte da Internet (UDP e TCP), bem como seus problemas de desempenho. Também serão estudadas questões relativas a redes sem fios.

O Capitulo 7 é dedicado à camada de aplicação, seus protocolos e suas aplicações. O primeiro tópico é o DNS, o catálogo telefônico da Internet. Em seguida, vem o correio eletrônico com uma discussão de seus protocolos. Depois, passamos para a Web, com descrições detalhadas do conteúdo estático, do conteúdo dinâmico, do que acontece no lado cliente, do que acontece no lado servidor, de protocolos, de desempenho, da Web sem fio e de vários outros temas. Finalmente, examinamos multimídia em rede, incluindo o fluxo de áudio, o rádio da Internet e o vídeo por demanda.

O Capítulo 8 se dedica à segurança das redes. Esse tópico tem aspectos que se relacionam a todas as camadas; assim, é mais fácil estudá-lo depois que todas as camadas tiverem sido completamente examinadas. O capítulo começa com uma introdução à criptografia. Mais adiante mostra como a criptografia pode ser usada para garantir a segurança da comunicação, do correio eletrônico e da Web. O livro termina com uma discussão de algumas áreas em que a segurança atinge a privacidade, a liberdade de expressão, a censura e outras questões sociais

conflitantes.

O Capítulo 9 contém uma lista comentada de leituras sugeridas, organizadas por capitulo. Seu objetivo é ajudar os leitores que desejam ter mais conhecimentos sobre redes. O capítulo também apresenta uma bibliografia em ordem alfabética com todas as referências citadas neste livro.

O Web site do autor na Prentice Hall é:

http://www.prenhall.com/tanenbaum

Ele contém uma página com links para muitos tutoriais, FAQs, empresas, consórcios industriais, organizações profissionais, organizações de padrões, tecnologias, documentos e muitos outros.

[T2] 1.9 Resumo

As redes de computadores podem ser usadas para inúmeros serviços, tanto por empresas quanto por indivíduos. Para as empresas, as redes de computadores pessoais que utilizam servidores compartilhados com freqüência oferecem acesso a informações corporativas. Para as pessoas, as redes oferecem acesso a uma série de informações e fontes de entretenimento. Em geral, as pessoas têm acesso à Internet fazendo uma ligação para um ISP com a utilização de um modem, embora um número cada vez maior de pessoas tenha uma conexão fixa em casa (pelo menos, nos EUA). Uma área de grande desenvolvimento atual é a das redes sem fios, com novas aplicações como acesso ao correio eletrônico em trânsito e m-commerce.

Grosso modo, as redes podem ser divididas em LANs, MANs, WANs e inter-redes, cada qual com suas próprias características, tecnologias, velocidades e nichos de mercado. As LANs abrangem um edifício e operam em altas velocidades. As MANs abrangem uma cidade, por exemplo, o sistema de televisão a cabo, que é utilizado hoje por muitas pessoas para acessar a Internet. As WANs abrangem um

país ou um continente. LANs e MANs são redes não comutadas (ou seja, não têm roteadores); as WANs são comutadas. As redes sem fios estão se tornando extremamente populares, em especial as LANs sem fios. As redes podem ser interconectadas para formar inter-redes.

O software de rede consiste em protocolos ou regras pelas quais os processos se comunicam. Os protocolos podem ser sem conexões ou orientados a conexões. A maioria das redes aceita as hierarquias de protocolos, com cada camada fornecendo serviços às camadas situadas acima dela e isolando-as dos detalhes dos protocolos usados nas camadas inferiores. Geralmente, as pilhas de protocolos se baseiam no modelo OSI ou no modelo TCP/IP. Esses dois modelos têm camadas de rede, transporte e aplicação, mas apresentam diferenças nas outras camadas. As questões de projeto incluem multiplexação, controle de fluxo, controle de erros e outras. Grande parte deste livro lida com protocolos e seu projeto.

As redes fornecem serviços a seus usuários. Esses serviços podem ser orientados a conexões ou sem conexões. Em algumas redes, o serviço sem conexões é fornecido em uma camada e o serviço orientado a conexões é oferecido na camada acima dela.

Entre as redes mais conhecidas estão a Internet, as redes ATM, a Ethernet e a LAN sem fio IEEE 802.11. A Internet evoluiu a partir da ARPANET, à qual foram acrescentadas outras redes para formar uma inter-rede. A Internet atual é na realidade um conjunto com muitos milhares de redes, e não uma única rede. O que a caracteriza é o uso da pilha de protocolos TCP/IP em toda sua extensão. O ATM é amplamente usado no sistema de telefonia para tráfego de dados de longa distância. A Ethernet é a LAN mais popular e está presente na maioria das grandes empresas e universidades. Por fim, as LANs sem fios a velocidades muito altas (até 54 Mbps) estão começando a ser extensamente desenvolvidas.

Fazer vários computadores se comunicarem exige uma extensa padronização, tanto de hardware quanto de software. Organizações como ITU-T, ISO, IEEE e IAB administram partes diferentes do processo de padronização.

[T2] Problemas

1. Imagine que você tenha treinado Bernie, seu cachorro São Bernardo, para carregar uma caixa de três fitas de 8 mm, em vez de um cantil de conhaque. (Quando seu disco ficar cheio, considere isso uma emergência.) Cada uma dessas fitas contém 7 gigabytes. O cachorro pode viajar a seu lado, onde quer que você esteja, a 18 km/h. Para que intervalo de distâncias Bernie terá uma taxa de dados mais alta que uma linha de transmissão cuja taxa de dados (excluindo o overhead) é de 150 Mbps?
2. Uma alternativa para uma LAN é simplesmente instalar um grande sistema de tempo compartilhado (timesharing) com terminais para todos os usuários. Apresente duas vantagens de um sistema cliente/servidor que utilize uma LAN.
3. O desempenho de um sistema cliente/servidor é influenciado por dois fatores de rede: a largura de banda da rede (quantos bits/s ela pode transportar) e a latência (quantos segundos o primeiro bit leva para ir do cliente até o servidor). Dê um exemplo de uma rede que exibe alta largura de banda e alta latência. Depois, dê um exemplo de uma rede com baixa largura de banda e baixa latência.
4. Além da largura de banda e da latência, que outro parâmetro é necessário para permitir uma boa caracterização da qualidade de serviço oferecida por uma rede empregada para tráfego de voz digitalizada?
5. Um fator que influencia no retardo de um sistema de comutação de pacotes store-and-forward é o tempo necessário para armazenar e encaminhar um pacote por um switch. Se o tempo de comutação é 10 μs, é provável que esse seja

um fator importante na resposta de um sistema cliente/servidor quando o cliente está em Nova York e o servidor está na Califórnia? Suponha que a velocidade de propagação em cobre e fibra seja igual a 2/3 da velocidade da luz no vácuo.

6. Um sistema cliente/servidor usa uma rede de satélite, com o satélite a uma altura de 40.000 km. Qual é a melhor retardo em resposta a uma solicitação?

7. No futuro, quando todo mundo tiver um terminal doméstico conectado a uma rede de computadores, será possível realizar plebiscitos instantâneos sobre questões importantes. É provável que a política atual seja eliminada, permitindo que as pessoas expressem seus desejos de uma maneira mais direta. Os aspectos positivos dessa democracia direta são óbvios; analise alguns dos aspectos negativos.

8. Um conjunto de cinco roteadores deve ser conectado a uma sub-rede ponto a ponto. Entre cada par de roteadores, os projetistas podem colocar uma linha de alta velocidade, uma linha de média velocidade, uma linha de baixa velocidade ou nenhuma linha. Se forem necessários 100 ms do tempo do computador para gerar e inspecionar cada topologia, quanto tempo será necessário para inspecionar todas elas?

9. Um grupo de $2^n - 1$ roteadores está interconectado em uma árvore binária centralizada, com um roteador em cada nó da árvore. O roteador i se comunica com o roteador j enviando uma mensagem à raiz da árvore. Em seguida, a raiz envia a mensagem de volta para j. Derive uma expressão aproximada para o número médio de hops (pulos) por mensagem para um grande valor de n, partindo do princípio de que todos os pares de roteadores são igualmente prováveis.

10. Uma desvantagem de uma sub-rede de difusão é a capacidade desperdiçada quando há vários hosts tentando acessar o canal ao mesmo tempo. Como um exemplo simples, suponha que o tempo esteja dividido em slots discretos, com

cada um dos *n* hosts tentando usar o canal com probabilidade *p* durante cada slot. Que fração dos slots é desperdiçada em conseqüência das colisões?

11. Quais são as duas razões para a utilização de protocolos dispostos em camadas?

12. O presidente da Specialty Paint Corp. resolve trabalhar com uma cervejaria local com a finalidade de produzir uma lata de cerveja invisível (como uma medida higiênica). O presidente pede que o departamento jurídico analise a questão e este, por sua vez, entra em contato com o departamento de engenharia. Como resultado, o engenheiro-chefe entra em contato com o funcionário de cargo equivalente na outra empresa para discutir os aspectos técnicos do projeto. Em seguida, os engenheiros enviam um relatório a seus respectivos departamentos jurídicos, que então discutem por telefone os aspectos legais. Por fim, os presidentes das duas empresas discutem as questões financeiras do negócio. Esse é um exemplo de protocolo em várias camadas no sentido utilizado pelo modelo OSI?

13. Qual é a principal diferença entre a comunicação sem conexão e a comunicação orientada a conexões?

14. Duas redes podem oferecer um serviço orientado a conexões bastante confiável. Uma delas oferece um fluxo de bytes confiável e a outra um fluxo de mensagens confiável. Elas são idênticas? Se forem, por que se faz essa distinção? Se não, dê um exemplo de como elas diferem.

15. O que significa "negociação" em uma discussão sobre protocolos de rede? Dê um exemplo.

16. Na Figura 1.19, é mostrado um serviço. Há outros serviços implícitos nessa figura? Nesse caso, onde? Se não, por que não?

17. Em algumas redes, a camada de enlace de dados trata os erros de transmissão solicitando a retransmissão dos quadros danificados. Se a

probabilidade de um quadro estar danificado é *p*, qual é o número médio de transmissões necessárias para enviar um quadro? Suponha que as confirmações nunca sejam perdidas.

18. Determine qual das camadas do modelo OSI trata de cada uma das tarefas a seguir:

(a) Dividir o fluxo de bits transmitidos em quadros.

(b) Definir a rota que será utilizada na sub-rede.

19. Se a unidade permutada no nível de enlace de dados for chamada quadro e a unidade permutada no nível de rede for chamada pacote, os quadros irão encapsular pacotes ou os pacotes irão encapsular os quadros? Explique sua resposta.

20. Um sistema tem uma hierarquia de protocolos com *n* camadas. As aplicações geram mensagens com *M* bytes de comprimento. Em cada uma das camadas, é acrescentado um cabeçalho com *h* bytes. Que fração da largura de banda da rede é preenchida pelos cabeçalhos?

21. Cite dois aspectos em que o modelo de referência OSI e o modelo de referência TCP/IP são iguais. Agora, cite dois aspectos em que eles são diferentes.

22. Qual é a principal diferença entre o TCP e o UDP?

23. A sub-rede da Figura 1.25(b) foi projetada para resistir a uma guerra nuclear. Quantas bombas seriam necessárias para particionar os nós em dois conjuntos desconectados? Suponha que qualquer bomba destrua um nó e todos os links conectados a ele.

24. A cada 18 meses, a Internet praticamente dobra de tamanho. Embora ninguém possa dizer com certeza, estima-se que havia 100 milhões de hosts em 2001. Use esses dados para calcular o número previsto de hosts da Internet em 2010. Você acredita nisso? Explique por que ou por que não.

25. Quando um arquivo é transferido entre dois computadores, são possíveis duas estratégias de confirmação. Na primeira, o arquivo é dividido em pacotes, que são confirmados individualmente pelo receptor, mas a transferência do arquivo como um todo não é confirmada. Na segunda, os pacotes não são confirmados individualmente mas, ao chegar a seu destino, o arquivo inteiro é confirmado. Analise essas duas abordagens.

26. Por que o ATM usa células pequenas de tamanho fixo?

27. Qual era o comprimento de um bit em metros no padrão 802.3 original? Utilize uma velocidade de transmissão de 10 Mbps e suponha que a velocidade de propagação no cabo coaxial seja igual a 2/3 da velocidade da luz no vácuo.

28. Uma imagem tem 1024 $\times$ 768 pixels com 3 bytes/pixel. Suponha que a imagem seja descompactada. Quanto tempo é necessário para transmiti-la por um canal de modem de 56 kbps? E por um modem a cabo de 1 Mbps? E por uma Ethernet de 10 Mbps? E pela Ethernet de 100 Mbps?

29. A Ethernet e as redes sem fios apresentam algumas semelhanças e algumas diferenças. Uma propriedade da Ethernet é a de que apenas um quadro pode ser transmitido de cada vez em uma rede Ethernet. A rede 802.11 compartilha essa propriedade com a Ethernet? Analise sua resposta.

30. As redes sem fios são fáceis de instalar, o que as torna econômicas, pois em geral os custos de instalação são muito maiores que os custos do equipamento. Apesar disso, elas também têm algumas desvantagens. Cite duas delas.

31. Liste duas vantagens e duas desvantagens da existência de padrões internacionais para protocolos de redes.

32. Quando um sistema tem uma parte permanente e uma parte removível (como uma unidade de CD-ROM e o CD-ROM), é importante que o sistema seja padronizado, de modo que empresas diferentes possam produzir as partes permanentes e as removíveis, para que elas sejam compatíveis entre si. Cite três

exemplos fora da indústria de informática em que esses padrões internacionais estão presentes. Agora, cite três áreas fora da indústria de informática em que eles não estão presentes.

33. Faça uma lista de atividades que você pratica todo dia em que são utilizadas redes de computadores. Como sua vida seria alterada se essas redes de repente fossem desativadas?

34. Descubra quais são as redes usadas em sua escola ou em seu local de trabalho. Descreva os tipos de redes, as topologias e os métodos de comutação utilizados.

35. O programa *ping* lhe permite enviar um pacote de teste a um dado local e verificar quanto tempo ele demora para ir e voltar. Tente usar o *ping* para ver quanto tempo ele demora para trafegar do local em que você está até vários locais conhecidos. A partir desses dados, represente o tempo de trânsito de mão única pela Internet como uma função da distância. É melhor usar universidades, pois a localização de seus servidores é conhecida com grande precisão. Por exemplo, *berkeley.edu* está em Berkeley, Califórnia, *mit.edu* está em Cambridge, Massachusetts, *vu.nl* está em Amsterdã, Holanda, *www.usyd.edu.au* está em Sydney, Austrália e *www.uct.ac.za* está em Cidade do Capa, África do Sul.

36. Vá até o Web site da IETF, *www.ietf.org*, ver o que eles estão fazendo. Escolha um projeto de que você goste e escreva um relatório de meia página sobre o problema e a solução proposta.

37. A padronização é muito importante no mundo das redes. A ITU e a ISO são as principais organizações oficiais de padronização. Vá até seus Web sites, *www.itu.org* e *www.iso.org*, respectivamente, e conheça seu trabalho de padronização. Escreva um breve relatório sobre os tipos de itens que foram padronizados.

38. A Internet é composta por um grande número de redes. Sua organização

determina a topologia da Internet. Uma quantidade considerável de informações sobre a topologia da Internet está disponível on-line. Use um mecanismo de procura para descobrir mais sobre a topologia da Internet e escreva um breve relatório resumindo suas descobertas.

[TA1]Capítulo

[TA2]2

[T1]A camada física

Neste capítulo, vamos analisar a camada mais baixa da hierarquia mostrada na Figura 1.24. Ela define as interfaces mecânica, elétrica e de sincronização para a rede. Inicialmente, faremos uma análise teórica da transmissão de dados, apenas para descobrir que a Mãe Natureza impõe uma série de limites sobre o que pode ser enviado por um canal.

Em seguida, discutiremos três meios de transmissão: guiado (fio de cobre e fibra óptica), sem fio (rádio terrestre) e satélite. Esse material fornecerá informações fundamentais sobre as principais tecnologias de transmissão usadas em redes modernas.

O restante do capítulo será dedicado a três exemplos de sistemas de comunicação usados na prática nas redes geograficamente distribuídas: o sistema de telefonia (fixa), o sistema de telefonia móvel (ou celular) e o sistema de televisão a cabo. Os três utilizam fibra óptica no backbone, mas são organizados de maneira diferente e empregam tecnologias distintas na conexão dos terminais.

[T2] 2.1 A base teórica da comunicação de dados

As informações podem ser transmitidas por fios, fazendo-se variar alguma propriedade física, como voltagem (tensão elétrica) ou corrente. Representando o valor dessa voltagem ou corrente como uma função de tempo com um valor único, $f(t)$, podemos criar um modelo para o comportamento do sinal e analisá-lo matematicamente. Essa análise será o assunto das próximas seções.

[T3] 2.1.1 Análise de Fourier

No início do Século XIX, o matemático francês Jean-Baptiste Fourier provou que qualquer função periódica razoavelmente estável, $g(t)$, com o período T pode ser construída como a soma de um número (possivelmente infinito) de senos e co-senos.

[Inserir equação do O.A. p. 86a] (2-1)

onde $f = 1/T$ é a freqüência fundamental, a_n e b_n são as amplitudes do seno e do co-seno dos n-ésimos **harmônicos** (termos) e c é uma constante. Essa decomposição é chamada **série de Fourier.** A partir da série de Fourier, a função pode ser reconstruída; ou seja, se o período T for conhecido e as amplitudes forem dadas, a função original do tempo poderá ser encontrada efetuando-se as somas da Equação (2-1).

Um sinal de dados com uma duração finita (como acontece com todos eles) pode ser tratado com base na premissa de que ele repete o mesmo padrão (ou seja, o intervalo de T a $2T$ é igual ao de 0 a T etc.).

As a_n amplitudes podem ser calculadas para qualquer $g(t)$ dada, multiplicando-se ambos os lados da Equação (2-1) por $\text{sen}(2\pi kft)$ e em seguida integrando-se de 0 a T. Como:

[Inserir equação do O.A. p. 86b]

(a) para

(b) sen

apenas um termo do somatório sobrevive: a_n. O somatório b_n desaparece completamente. Da mesma forma, multiplicando a Equação (2-1) por $\cos(2\pi kft)$ e integrando entre 0 e T, podemos derivar b_n. Integrando ambos os lados da equação tal qual ela se encontra, podemos encontra c. Os resultados dessas operações são:

[Inserir equação do O.A. p. 86c]

sen

[T3] 2.1.2 Sinais limitados pela largura de banda

Para perceber como tudo isso se relaciona com a comunicação de dados, vamos analisar um exemplo específico: a transmissão do caractere ASCII "b" codificado como um byte de 8 bits. O padrão de bits que deve ser transmitido é 01100010. A parte esquerda da Figura 2.1(a) mostra a saída de voltagem do computador transmissor. A análise de Fourier desse sinal produz os seguintes coeficientes:

[Inserir equação do O.A. p. 86d]

(a) sen

[arte: ver original p. 87]

[Dísticos]

[1]Tempo

[2]Amplitude rms

[3]Número harmônico

[4]0,50

0,25

[5]1 harmônico

[6]2 harmônicos

[7]4 harmônicos

[8]8 harmônicos

[9]Tempo

[10]Número de harmônico

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 2.1

[FL] (a) Um sinal binário e suas amplitudes de média quadrática de Fourier. (b) a (e) Aproximações sucessivas do sinal original

As amplitudes de média quadrática, [ver símbolo], para os primeiros termos são mostradas no lado direito da Figura 2.1 (a). Esses valores têm interesse porque seus quadrados são proporcionais à energia transmitida na freqüência correspondente.

Nenhum recurso de transmissão é capaz de transmitir sinais sem perder parte da energia no processo. Se todos os coeficientes da série de Fourier fossem igualmente reduzidos, o sinal resultante seria reduzido em amplitude, mas não seria distorcido [isto é, ele teria a mesma forma mostrada na Figura 2.1(a)]. Infelizmente, todos os meios de transmissão reduzem diferentes componentes de Fourier por diferentes valores e, em conseqüência disso, introduzem distorção. Em geral, as amplitudes são transmitidas sem redução, de 0 até alguma freqüência f_c [medida em ciclos/s ou Hertz (Hz)], com todas as freqüências acima dessa freqüência de corte (cutoff frequency) sendo atenuadas. A faixa de freqüências transmitidas sem serem fortemente atenuadas denomina-se **largura de banda.** Na prática, o corte não é nítido; assim, muitas vezes a largura de banda varia desde 0 até a freqüência em que metade da potência é transmitida.

A largura de banda é uma propriedade física do meio de transmissão, e em geral depende da construção, da espessura e do comprimento do meio. Em alguns casos um filtro é introduzido no circuito para limitar o volume de largura de banda disponível para cada cliente. Por exemplo, uma linha telefônica pode ter uma largura de banda de 1 MHz para curtas distâncias, mas as empresas de telefonia acrescentam um filtro que restringe cada cliente a cerca de 3100 Hz. Essa largura de banda é adequada para voz compreensível e melhora a eficiência do sistema, limitando a utilização de recursos pelos clientes.

Vejamos agora como seria a forma do sinal da Figura 2.1(a) se a largura de banda fosse tão baixa que apenas as freqüências mais baixas fossem transmitidas [ou seja, se a função estivesse sendo aproximada pelos primeiros termos da Equação (2-1)]. A Figura 2.1(b) mostra o sinal resultante de um canal pelo qual apenas o primeiro harmônico (o fundamental, *f*) pode passar. Da mesma forma, a Figura 2.1 (c)-(e) mostra os espectros e as funções reconstruídas para canais com uma largura de banda mais alta.

Dada uma taxa de bits igual a *b* bits/s, o tempo necessário para o envio de 8 bits (por exemplo), um bit de cada vez, é 8/*b*s; sendo assim, a freqüência do primeiro harmônico é *b*/8 Hz. Uma linha telefônica comum, freqüentemente chamada **linha de qualidade de voz**, tem uma freqüência de corte artificialmente introduzida, pouco acima de 3000 Hz. Essa restrição significa que o número do harmônico mais alto transmitido é aproximadamente 3000/(*b*/8) ou 24.000/*b* (o corte não é preciso).

Para algumas taxas de dados, os números funcionam de acordo com o padrão mostrado na Figura 2.2. Esses números deixam claro que, quando se tenta fazer uma transmissão a 9.600 bps usando uma linha telefônica de qualidade de voz, o modelo sugerido na Figura 2.1(a) assume a forma mostrada na Figura 2.1 (c), tornando complicada a recepção precisa do fluxo original de bits. Podemos perceber também que, em taxas de dados bem mais altas que 38,4 kbps, não existe a menor possibilidade de que todos os sinais sejam *binários*, mesmo quando não há o menor ruído no meio de transmissão. Em outras palavras, limitando-se a largura de banda, limita-se a taxa de dados, mesmo nos canais perfeitos (sem ruídos). No entanto, sofisticados esquemas de codificação que usam diversos níveis de voltagem possibilitam a existência e a utilização de taxas de dados mais altas. Vamos discutir essa questão ao longo deste capítulo.

[arte: ver original p. 89]

[T]Tabela

Bps	T(ms)	Primeiro harmônico (Hz)	Núm. de harmônicos enviados
300	26,67	37,5	80
600	13,33	75	40
1200	6,67	150	20
2400	3,33	300	10
4800	1,67	600	5
9600	0,83	1200	2
19200	0,42	2400	1
38400	0,21	4800	0

[F]Figura 2.2

[FL] Relação entre as taxas de dados e os harmônicos

[T3] 2.1.3 Taxa máxima de dados de um canal

Em 1924, H. Nyquist, um engenheiro da AT&T, percebeu que até mesmo um canal perfeito tem uma capacidade de transmissão finita. Ele derivou uma equação expressando a taxa máxima de dados de um canal sem ruído com largura de banda finita. Em 1948, Claude Shannon aprofundou o trabalho de Nyquist e o estendeu ao caso de um canal sujeito a ruído aleatório (isto é, termodinâmico) (Shannon, 1948). Veja a seguir um resumo dos resultados, agora clássicos, dessas experiências.

Nyquist provou que, se um sinal arbitrário atravessar um filtro com baixa freqüência de corte H, o sinal filtrado poderá ser completamente reconstruído a partir de apenas $2H$ amostras (exatas) por segundo. Fazer uma amostragem da linha com uma rapidez maior que $2H$ vezes por segundo seria inútil, pois os componentes de freqüência mais alta que essa amostragem poderia recuperar já teriam sido filtrados. Se o sinal consistir em V níveis discretos, o teorema de

Nyquist afirma que:

taxa máxima de dados = $2H \log_2 V$ bits/s

Por exemplo, um canal de 3 kHz sem ruído não pode transmitir sinais binários (ou seja, de dois níveis) a uma taxa superior a 6000 bps.

Até agora, só mencionamos os canais sem ruído. Se houver ruído aleatório, a situação irá se deteriorar com rapidez. Além disso, sempre existe ruído aleatório (térmico) presente, devido ao movimento das moléculas no sistema. O volume de ruído térmico presente é medido pela relação entre a potência do sinal e a potência do ruído, chamada relação **sinal/ruído.** Se representarmos a potência do sinal por S e a potência do ruído por N, a relação sinal/ruído será S/N. Em geral, não se faz referência à relação propriamente dita; em vez disso, utiliza-se a quantidade $10 \log_{10} S/N$. Essas unidades são chamadas **decibéis** (dB). Uma relação S/N igual a 10 corresponde a 10 dB, uma relação igual a 100 equivale a 20 dB, uma relação igual a 1000 equivale a 30 dB e assim por diante. Com freqüência, os fabricantes de amplificadores estereofônicos caracterizam a largura de banda (faixa de freqüência) na qual seu produto é linear oferecendo a freqüência de 3 dB em cada extremidade. Esses são os pontos em que o fator de amplificação foi dividido aproximadamente ao meio (porque [ver símbolo]).

O principal resultado de Shannon é que a taxa máxima de dados de um canal com ruídos cuja largura de banda é H Hz, e cuja relação sinal/ruído é S/N, é dada por:

número máximo de bits/s = $H \log_2 (1 + S/N)$

Por exemplo, um canal de largura de banda 3000 Hz com uma relação de sinal para ruído térmico igual a 30 dB (parâmetros típicos da parte analógica do sistema telefônico) nunca pode transmitir muito mais de 30000 bps, independente da quantidade de níveis de sinal utilizados e da freqüência com que as amostras são obtidas. O resultado de Shannon utilizou os argumentos da teoria da informação e se aplica a qualquer canal sujeito a ruído térmico. Os

exemplos opostos devem ser tratados na mesma categoria das máquinas de movimento contínuo (moto perpétuo). Deve-se observar que esse é apenas um limite superior, raramente alcançado pelos sistemas reais.

[T2] 2.2 Meios de transmissão guiados

O objetivo da camada física é transmitir um fluxo bruto de bits de uma máquina para outra. Vários meios físicos podem ser usados para realizar a transmissão. Cada um tem seu próprio nicho em termos de largura de banda, retardo, custo e facilidade de instalação e manutenção. Os meios físicos são agrupados em meios guiados, como fios de cobre e fibras ópticas, e em meios não guiados, como as ondas de rádio e os raios laser transmitidos pelo ar. Discutiremos esses meios de transmissão nas próximas seções.

[T3] 2.2.1 Meios magnéticos

Uma das formas mais comuns de transportar dados de um computador para outro é gravá-los em fita magnética ou em mídia removível (por exemplo, DVDs graváveis), transportar fisicamente a fita ou os discos para a máquina de destino, onde eles finalmente serão lidos. Apesar de não ser tão sofisticado quanto a utilização de um satélite de comunicações geossíncrono, esse método costuma ser muito mais eficaz sob o ponto de vista financeiro, em especial nas aplicações em que a alta largura de banda ou o custo por bit tem importância fundamental. Basta fazer um cálculo simples para esclarecer essa questão. Uma fita Ultrium de padrão industrial pode armazenar 200 gigabytes. Uma caixa de 60 × 60 × 60 cm pode conter cerca de 1000 fitas desse tipo, perfazendo uma capacidade total de 200 terabytes, ou 1600 terabits (1,6 petabits). Uma caixa de fitas pode ser entregue em qualquer parte do país em 24 horas pelo serviço Sedex dos Correios, pela Federal Express e por outras transportadoras. A largura de banda efetiva

dessa transmissão é de 1600 terabits/86400 s, ou 19 Gbps. Se o destino estiver a apenas uma hora de distância, a largura de banda será ampliada para mais de 400 Gbps. Nenhuma rede de computadores consegue nem mesmo se aproximar desse desempenho.

No caso de um banco com muitos gigabytes de dados a serem gravados em uma segunda máquina por meio de uma operação diária de backup (de modo que o banco possa continuar a funcionar mesmo que ocorra uma grande enchente ou um terremoto), é provável que nenhuma outra tecnologia de transmissão possa sequer ser comparada à fita magnética, em termos de desempenho. É claro que as redes estão ficando mais rápidas, mas as densidades das fitas também estão aumentando.

Se consideramos o custo, obteremos um quadro semelhante. O custo de uma fita Ultrium é aproximadamente $ 40 quando a compra é feita no atacado. Uma fita pode ser reutilizada pelo menos 10 vezes. Portanto, o custo das fitas passa a ser $ 4000 por caixa, para cada utilização. Adicione a esse montante mais $ 1000 pelo frete (provavelmente muito menos) e teremos um custo final aproximado de $ 5000 para transportar 200 TB. Conseqüentemente, gastaremos para transportar 1 gigabyte menos de 3 centavos de dólar. Nenhuma rede pode competir com esses valores. Moral da história:

Nunca subestime a largura de banda de uma caminhonete cheia de fitas "voando" na estrada.

[T3] 2.2.2 Par trançado

Embora as características de largura de banda da fita magnética sejam excelentes, as características de retardo são ruins. O tempo de transmissão é medido em minutos ou horas, e não em milissegundos. Muitas aplicações precisam de uma conexão on-line. O meio de transmissão mais antigo e ainda

mais comum é o **par trançado**. Um par trançado consiste em dois fios de cobre encapados, que em geral têm cerca de 1 mm de espessura. Os fios são enrolados de forma helicoidal, assim como uma molécula de DNA. O trançado dos fios é feito porque dois fios paralelos formam uma antena simples. Quando os fios são trançados, as ondas de diferentes partes dos fios se cancelam, o que significa menor interferência.

A aplicação mais comum do par trançado é o sistema telefônico. Quase todos os telefones estão conectados à estação central da companhia telefônica por um par trançado. Os pares trançados podem se estender por diversos quilômetros sem amplificação mas, quando se trata de distâncias mais longas, existe a necessidade de repetidores. Quando muitos pares trançados percorrem paralelamente uma distância muito grande, como acontece na ligação entre um prédio e a estação central da companhia telefônica, eles são envolvidos por uma capa protetora. Se não estivessem trançados, esses pares provocariam muitas interferências. Nos países em que as linhas telefônicas são instaladas em postes, com freqüência vemos cabos de pares trançados com vários centímetros de diâmetro.

Os pares trançados podem ser usados na transmissão de sinais analógicos ou digitais. A largura de banda depende da espessura do fio e da distância percorrida mas, em muitos casos, é possível alcançar diversos megabits/s por alguns quilômetros. Devido ao custo e ao desempenho obtidos, os pares trançados são usados em larga escala e é provável que assim permaneçam nos próximos anos.

Existem diversos tipos de cabeamento de pares trançados, dois dos quais são importantes para as redes de computadores. Os pares trançados da **categoria 3** consistem em dois fios encapados cuidadosamente trançados. Em geral, quatro pares desse tipo são agrupados dentro de uma capa plástica protetora, onde os

fios são mantidos juntos. Até 1988, a maioria dos prédios tinha um cabo da categoria 3 ligando cada um dos escritórios a um **armário de fiação** (ou **gabinete de fiação**) em cada andar. Esse esquema permitia que até quatro telefones normais ou dois telefones de várias linhas em cada escritório se conectassem ao equipamento da companhia telefônica instalado no armário de fiação.

Em 1988, foram lançados os pares trançados da **categoria 5**, mais avançados. Eles eram parecidos com os pares da categoria 3, mas tinham mais voltas por centímetro, o que resultou em menor incidência de linhas cruzadas e em um sinal de melhor qualidade nas transmissões de longa distância; isso os tornou ideais para a comunicação de computadores de alta velocidade. Estão sendo lançadas as categorias 6 e 7, capazes de tratar sinais com largura de banda de 250 MHz e 600 MHz, respectivamente (em comparação com apenas 16 MHz e 100 MHz para as categorias 3 e 5, respectivamente).

Todos esses tipos de fiação costumam ser chamados **UTP (Unshielded Twisted Pair)**, para diferenciá-los dos volumosos e caros cabos trançados IBM lançados no início da década de 1980 que, no entanto, não se mostraram populares fora das instalações IBM. O cabeamento de par trançado está ilustrado na Figura 2.3.

[arte: ver original p. 92]

[Dísticos]

[1] (a) (b)

[F]Figura 2.3

[FL](a) UTP da categoria 3. (b) UTP da categoria 5

[T3] 2.2.3 Cabo coaxial

Outro meio de transmissão comum é o **cabo coaxial** (conhecido apenas como "coax"). Ele tem melhor blindagem que os pares trançados, e assim pode se estender por distâncias mais longas em velocidades mais altas. Dois tipos de

cabo coaxial são amplamente utilizados. Um deles, o cabo de 50 ohms, é comumente empregado nas transmissões digitais. O outro tipo, o cabo de 75 ohms, é usado com freqüência nas transmissões analógicas e de televisão a cabo, mas está se tornando mais importante com o advento da Internet por cabo. Essa distinção se baseia mais em fatores históricos do que técnicos (por exemplo, as primeiras antenas dipolo tinham uma impedância de 300 ohms e era fácil desenvolver transformadores de casamento de impedância de 4:1).

Um cabo coaxial consiste em um fio de cobre esticado na parte central, envolvido por um material isolante. O isolante é protegido por um condutor cilíndrico, geralmente uma malha sólida entrelaçada. O condutor externo é coberto por uma camada plástica protetora. A Figura 2.4 apresenta uma vista em corte de um cabo coaxial.

[arte: ver original p. 93]

[Dísticos]

[1]Núcleo de cobre

[2]Material isolante

[3]Condutor externo em malha

[4]Capa plástica protetora

[F]Figura 2.4

[FL] Um cabo coaxial

A construção e a blindagem do cabo coaxial proporcionam a ele uma boa combinação de alta largura de banda e excelente imunidade a ruído. A largura de banda possível depende da qualidade do cabo, do tamanho e da relação sinal/ruído do sinal de dados. Os cabos modernos têm uma largura de banda próxima de 1 GHz. Os cabos coaxiais eram muito usados no sistema telefônico em linhas de longa distância, mas agora estão sendo substituídos por fibras

ópticas nas rotas de longa distância. Porém, os cabos coaxiais ainda são usados em larga escala pelas redes de televisão a cabo e em redes metropolitanas.

[T3] 2.2.4 Fibra óptica

Muitas pessoas da indústria de informática se orgulham da rapidez com que a tecnologia usada nos computadores vem melhorando. O IBM PC original (de 1981) funcionava com uma velocidade do clock de 4,77 MHz. Vinte anos depois, os PCs podiam funcionar a 2 GHz, um aumento de desempenho de vinte vezes por década. Nada mau.

No mesmo período, a comunicação de dados geograficamente distribuída passou de 56 Kbps (a ARPANET) para 1 Gbps (comunicação óptica moderna); isso significa que seu desempenho melhorou mais de 125 vezes em cada década, enquanto no mesmo período a taxa de erros passou de 10^{-5} por bit para quase zero.

Alem disso, as CPUs isoladas estão se aproximando dos limites físicos, como a velocidade da luz e os problemas decorrentes da dissipação de calor. Por outro lado, com a *atual* tecnologia de fibra óptica, a largura de banda pode ultrapassar a casa dos 50.000 Gbps (50 Tbps) e muitas pessoas estão procurando tecnologias e materiais de melhor qualidade. O limite prático da sinalização atual é de cerca de 10 Gbps, devido à nossa incapacidade para realizar a conversão entre sinais elétricos e ópticos com velocidade maior, embora já se tenha alcançado em laboratório a velocidade de 100 Gbps em uma única fibra.

Na corrida entre informática e comunicação, a comunicação ganhou. As implicações reais de uma largura de banda essencialmente infinita (apesar de não ter custo zero) ainda não foram totalmente assimiladas por uma geração de cientistas e engenheiros da computação que aprenderam a pensar em termos dos baixos limites de Nyquist e Shannon impostos pelo fio de cobre. Os novos

conceitos partem da premissa de que todos os computadores são terrivelmente lentos e, por essa razão, as redes devem tentar evitar a computação a todo custo, independente do desperdício de largura de banda que isso signifique. Nesta seção, vamos estudar as fibras ópticas e veremos como funciona essa tecnologia de transmissão.

Um sistema de transmissão óptica tem três componentes fundamentais: a fonte de luz, o meio de transmissão e o detector. Por convenção, um pulso de luz indica um bit 1, e a ausência de luz representa um bit zero. O meio de transmissão é uma fibra de vidro ultrafina. O detector gera um pulso elétrico quando entra em contato com a luz. Quando instalamos uma fonte de luz em uma extremidade de uma fibra óptica e um detector na outra, temos um sistema de transmissão de dados unidirecional que aceita um sinal elétrico, converte o sinal e o transmite por pulsos de luz; depois, na extremidade de recepção, a saída é reconvertida em um sinal elétrico.

Esse sistema de transmissão desperdiçaria luz e na prática não teria a menor utilidade, exceto como um interessante princípio físico. Quando um raio de luz passa de um meio para outro — por exemplo, de sílica fundida para o ar — o raio é refratado (desviado) na fronteira sílica/ar, como mostra a Figura 2.5 (a). Na figura, vemos um raio de luz incidente na fronteira em um ângulo α_1, emergindo em um ângulo β_1. A intensidade da refração depende das propriedades dos dois meios físicos (em particular, de seus índices de refração). Para ângulos de incidência que ultrapassam um certo valor crítico, a luz é refletida de volta para a sílica; nada escapa para o ar. Dessa forma, um raio de luz incidente no ângulo crítico ou acima dele é interceptado no interior da fibra, como mostra a Figura 2.5(b), e pode se propagar por muitos quilômetros sem sofrer praticamente nenhuma perda.

[arte: ver original p. 94]

[Dísticos]

[1]Ar

β_1 β_2 β_3

Fronteira ar/sílica

α_1 α_2 α_3

Sílica

(a)

[2]Reflexão total interna

Fonte de luz

(b)

[F]Figura 2.5

[FL] (a) Três exemplos de um raio de luz dentro de uma fibra de sílica incidindo na fronteira ar/sílica em diferentes ângulos. (b) A luz interceptada pela reflexão total interna

O exemplo da Figura 2.5(b) mostra apenas um raio interceptado mas, como qualquer raio de luz incidente na fronteira acima do ângulo crítico será refletido internamente, muitos raios distintos estarão ricocheteando em diferentes ângulos. Dizemos que cada raio tem um **modo** específico; assim, uma fibra que apresenta essa propriedade é chamada **fibra multimodo.**

No entanto, se o diâmetro da fibra for reduzido a alguns comprimentos de onda de luz, a fibra agirá como um guia de onda, e a luz só poderá se propagar em linha reta, sem ricochetear, produzindo assim uma **fibra de modo único** ou **fibra monomodo.** As fibras de modo único são mais caras, mas são amplamente utilizadas em distâncias mais longas. As fibras de modo único disponíveis no momento podem transmitir dados a 50 Gbps por 100 km sem amplificação. Foram obtidas taxas de dados ainda mais altas no laboratório, para distâncias

mais curtas.

[T4] Transmissão de luz na fibra

As fibras ópticas são feitas de vidro que, por sua vez, é produzido a partir da areia, uma matéria-prima de baixo custo e abundante. Os antigos egípcios já dominavam a manufatura do vidro, mas o vidro produzido por eles não podia ter mais de 1 mm de espessura, para que a luz pudesse atravessá-lo. O vidro transparente usado nas janelas foi desenvolvido durante a Renascença. O vidro usado nas modernas fibras ópticas é tão transparente que, se em vez de água os oceanos fossem cheios desse tipo vidro, seria possível ver o fundo do mar da superfície, da mesma forma que é possível ver o solo quando voamos de avião em um dia ensolarado.

A atenuação da luz através do vidro depende do comprimento de onda da luz (bem como de algumas propriedades físicas do vidro). A atenuação do tipo de vidro usado nas fibras é mostrada na Figura 2.6 em decibéis por quilômetro linear de fibra. A atenuação em decibéis é obtida pela seguinte fórmula:

[Inserir equação do O.A. p. 95]

(a) Atenuação em decibéis = 10 $\log_{10}$

(b) potência transmitida

(c) potência recebida

Por exemplo, quando o fator de perda é igual a dois, obtemos a atenuação de 10 $\log_{10} 2 = 3$ dB. A figura mostra a parte de infravermelho do espectro que, na prática, é a utilizada. A luz visível tem comprimentos de onda ligeiramente mais curtos, que variam de 0,4 a 0,7 mícron (1 mícron é igual a 10^{-6} metros). Na verdade, esses comprimentos de onda seriam de 400 nm e 700 nm, mas vamos manter a nomenclatura tradicional.

A comunicação óptica utiliza três bandas de comprimentos de onda. Elas são

centralizadas em 0,85, 1,30 e 1,55 micra, respectivamente. As duas últimas têm boas propriedades de atenuação (uma perda inferior a 5% por quilômetro). A banda de 0,85 mícron tem uma atenuação maior mas, por outro lado, nesse comprimento de onda, os lasers e os chips podem ser produzidos a partir do mesmo material (arsenieto de gálio). As três bandas têm entre 25.000 e 30.000 GHz de largura.

[arte: ver original p. 95]

[Dísticos]

[1]Atenuação (dB/km)

2,0

1,8

1,6

1,4

1,2

1,0

0,8

0,6

0,4

0,2

[2]Banda de 0,85µ

[3]Banda de 1,30 µ

[4]Banda de 1,55 µ

[5]0 0,8 0,9 1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1,8

[6]Comprimento de onda (micra)

[F]Figura 2.6

[FL] Atenuação da luz na fibra, na região do infravermelho

Os pulsos de luz enviados através de uma fibra se expandem à medida que se propagam. Essa expansão é chamada **dispersão cromática.** O volume da dispersão depende do comprimento de onda. Uma forma de impedir que esses pulsos dispersos se sobreponham é aumentar a distância entre eles, mas isso só pode ser feito reduzindo-se a taxa de sinalização. Felizmente, descobriu-se que, quando os pulsos são produzidos em uma forma especial relacionada ao recíproco do co-seno hiperbólico, todos os efeitos de dispersão são cancelados e é possível enviar pulsos por milhares de quilômetros sem que haja uma distorção significativa. Esses pulsos são chamados **solitons.** Atualmente, o mundo assiste a um grande esforço de pesquisa no sentido de colocar em prática as experiências que estão sendo feitas em laboratórios com os solitons.

[T4] Cabos de fibra

Os cabos de fibra óptica são semelhantes aos cabos coaxiais, exceto por não terem a malha metálica. A Figura 2.7(a) mostra a vista lateral de uma única fibra. No centro, fica o núcleo de vidro através do qual se propaga a luz. Nas fibras multimodo, o núcleo tem 50 micra de diâmetro, o que corresponde à espessura de um fio de cabelo humano. Nas fibras monomodo, o núcleo tem entre 8 e 10 micra.

[arte: ver original p. 96]

[Dísticos]

[1]Núcleo (vidro)

[2] Revestimento interno (vidro)

[3] Cobertura (plástico)

[4] (a)

[5] Revestimento

[6] Cobertura

[7] Revestimento interno

[8] (b)

[9] Núcleo

[F]Figura 2.7

[FL] (a) Vista lateral de uma única fibra. (b) Vista da extremidade de um cabo com três fibras

O núcleo é envolvido por um revestimento de vidro com um índice de refração inferior ao do núcleo, para manter toda a luz no núcleo. Em seguida, há uma cobertura de plástico fino para proteger o revestimento interno. Geralmente, as fibras são agrupadas em feixes, protegidas por um revestimento exterior. A Figura 2.7(b) mostra um cabo com três fibras.

Normalmente, os cabos de fibra terrestres são colocadas no solo a um metro da superfície, onde ocasionalmente são atacados por pequenos animais roedores. Próximo ao litoral, cabos de fibra transoceânicos são enterrados em trincheiras por uma espécie de arado marítimo. Em águas profundas, eles são depositados no fundo, onde podem ser arrastados por redes de pesca ou comidos por tubarões.

As fibras podem estar conectadas de três maneiras diferentes. Em primeiro lugar, elas podem ter conectores em suas extremidades e serem plugadas em soquetes de fibra. Os conectores perdem de 10 a 20% da luz, mas facilitam a reconfiguração dos sistemas.

Em segundo lugar, elas podem ser unidas mecanicamente. Nesse caso, as duas extremidades são cuidadosamente colocadas uma perto da outra em uma luva especial e fixadas no lugar. O alinhamento pode ser melhorado fazendo-se a luz passar pela junção e, em seguida, realizando-se pequenos ajustes cuja finalidade é maximizar o sinal. As junções mecânicas são realizadas em cerca de 5 minutos

por uma equipe treinada e resultam em uma perda de 10% da luz.

Em terceiro lugar, dois peças de fibra podem ser fundidas de modo a formar uma conexão sólida. A união por fusão é quase tão boa quanto uma fibra sem emendas; no entanto, mesmo nesse caso, há uma pequena atenuação.

Nos três tipos de uniões podem ocorrer reflexões no ponto de junção, e a energia refletida pode interferir com o sinal.

Dois tipos de fontes de luz são usadas geralmente para fazer a sinalização: os diodos emissores de luz (LEDs — Light Emitting Diodes) e os lasers semicondutores. Eles têm diferentes propriedades, como mostra a Figura 2.8. O comprimento de onda desses elementos pode ser ajustado pela inserção de interferômetros de Fabry-Perot ou Mach-Zehnder entre a fonte e a fibra. Os interferômetros de Fabry-Perot são cavidades ressonantes simples que consistem em dois espelhos paralelos. A luz incide perpendicularmente aos espelhos. O comprimento da cavidade filtra os comprimentos de onda que cabem em um número inteiro de períodos. Os interferômetros de Mach-Zehnder separam a luz em dois feixes. Os dois feixes percorrem distâncias ligeiramente diferentes. Eles são recombinados no destino e só ficam em fase para certos comprimentos de onda.

[arte: ver original p. 97]

[T]Tabela

Item	**LED**	**Laser semicondutor**
Taxa de dados	Baixa	Alta
Tipo de fibra	Multimodo	Multimodo ou modo único
Distância	Curta	Longa
Duração	Longa	Curta
Sensibilidade à temperatura	Insignificante	Substancial
Custo	Baixo custo	Dispendioso

[F]Figura 2.8

[FL] Uma comparação entre diodos semicondutores e LEDs utilizados como fontes de luz

A extremidade de recepção de uma fibra óptica consiste em um fotodiodo, que emite um pulso elétrico ao ser atingido pela luz. Em geral, o tempo de resposta de um fotodiodo é 1 nanossegundo, o que limita as taxas de dados a 1 Gbps. O ruído térmico também é importante, pois um pulso de luz deve conduzir energia suficiente para ser detectado. Com pulsos de potência suficiente, a taxa de erros pode se tornar arbitrariamente pequena.

[T4] Redes de fibra óptica

As fibras ópticas podem ser usadas em LANs e nas transmissões de longa distância, apesar de sua conexão ser mais complexa que a conexão a uma rede Ethernet. Uma forma de contornar esse problema é perceber que uma rede em anel é, na verdade, apenas um conjunto de enlaces ponto a ponto, como mostra a Figura 2.9. A interface de cada computador percorre o fluxo de pulsos de luz até o próximo enlace e também serve como junção em forma de T para permitir que o computador envie e aceite mensagens.

São usados dois tipos de interfaces. Uma interface passiva consiste em dois conectores fundidos à fibra principal. Um conector tem um LED ou um diodo a laser na sua extremidade (para transmissão), e o outro tem um fotodiodo (para recepção). O conector em si é completamente passivo e, por essa razão, é bastante confiável, pois um LED ou um fotodiodo quebrado não compromete o anel. No máximo, ele deixa um computador off-line.

[arte: ver original p. 98]

[Dísticos]

[1] Computador

[2] Detalhe de interface

[3] De/para computador

[4] Fio de cobre

[5] Direção de propagação da luz

[6] Transmissor óptico (LED)

[7]Regenerador de sinal (elétrico)

[8] Receptor óptico (fotodiodo)

[9] Fibra

[10] Interface

[11] Fibra óptica

[F]Figura 2.9

[FL] Um anel de fibra óptica com repetidores ativos

O outro tipo de interface, mostrado na Figura 2.9, é o **repetidor ativo**. A luz recebida é convertida em um sinal elétrico, tem sua capacidade regenerada caso ela tenha sido enfraquecida e é retransmitida na forma de luz. A interface com o computador é um fio de cobre comum que passa pelo regenerador de sinal. Já estão sendo usados repetidores puramente ópticos. Esses dispositivos dispensam as conversões ópticas/elétricas/ópticas; isso significa que eles podem operar em larguras de banda extremamente altas.

Se um repetidor ativo entrar em pane, o anel será interrompido e a rede, desfeita. Por outro lado, como o sinal é regenerado em cada interface, os enlaces individuais entre computadores podem se estender por quilômetros, praticamente sem limite sobre o tamanho total do anel. As interfaces passivas perdem luz em cada junção; por isso, o número total de computadores e o tamanho total do anel acabam sofrendo grandes restrições.

Uma topologia em anel não é a única forma de se construir uma LAN usando fibras ópticas. Também é possível ter um hardware se comunicando por difusão, com o uso da construção em **estrela passiva** da Figura 2.10. Nesse projeto, cada interface tem uma fibra entre seu transmissor e um cilindro de sílica, e as fibras de entrada são fundidas em uma extremidade do cilindro. Da mesma forma, as fibras fundidas à outra extremidade do cilindro são conectadas a cada um dos receptores. Sempre que uma interface emite um pulso de luz, ele é difundido dentro da estrela passiva para iluminar todos os receptores e, dessa forma, possibilitar a difusão dos dados. Na verdade, a estrela passiva combina todos os sinais de entrada e transmite o resultado da fusão em todas as linhas. Como a energia de entrada é dividida entre todas as linhas de saída, o número de nós na rede é limitado pela sensibilidade dos fotodiodos.

[T4] Comparação entre fibras ópticas e fios de cobre

É instrutivo comparar a fibra com o cobre. A fibra tem muitas vantagens. Para começar, ela pode gerenciar larguras de banda muito mais altas do que o cobre. Apenas essa característica justificaria seu uso nas redes de última geração. Devido à baixa atenuação, os repetidores só são necessários a cada 50 quilômetros de distância em linhas longas, comparada à distância de 5 km no caso do cobre, uma economia de custo significativa. A fibra também tem a vantagem de não ser afetada por picos de voltagem, interferência eletromagnética ou quedas no fornecimento de energia. Ela também está imune à ação corrosiva de alguns elementos químicos que pairam no ar e, conseqüentemente, adapta-se muito bem a ambientes industriais desfavoráveis.

[arte: ver original p. 99]

[Dísticos]

[1]Transmissor

[2]Receptor

[3] Interfaces de computadores

[4] Cada fibra de saída recebe a luz de todas as fibras de entrada

[5] Cada fibra de entrada ilumina toda a estrela

[F]Figura 2.10

[FL] Uma conexão de estrela passiva em uma rede de fibra óptica

Por mais estranho que possa parecer, as empresas telefônicas gostam da fibra por outra razão: ela é fina e leve. Muitos dos dutos de cabos atuais estão completamente lotados, de modo que não há espaço para aumentar sua capacidade. Além da remoção, e subseqüente substituição de todo o cobre por fibras esvaziar os dutos, o cobre tem um excelente valor de revenda para as refinarias especializadas, pois trata-se de um minério de altíssima qualidade. Além disso, a fibra é muito mais leve que o cobre. Mil pares trançados com 1 km de comprimento pesam 8 toneladas. Duas fibras têm mais capacidade e pesam apenas 100 kg, reduzindo de maneira significativa a necessidade de sistemas mecânicos de suporte, cuja manutenção é extremamente cara. Nas novas rotas, as fibras são preferidas, por terem um custo de instalação muito mais baixo. Por fim, as fibras não desperdiçam luz e dificilmente são interceptadas. Por essas razões, a fibra é uma alternativa com um excelente nível de segurança contra possíveis escutas telefônicas.

No entanto, a fibra tem a desvantagem de ser uma tecnologia menos familiar, exigindo conhecimentos que nem todos os engenheiros possuem e, além disso, as fibras podem ser danificadas com facilidade, se forem encurvadas demais. Como a transmissão óptica é basicamente unidirecional, a comunicação bidirecional exige duas fibras ou duas bandas de freqüência em uma única fibra. Por fim, as interfaces de fibra são mais caras que as interfaces elétricas. Apesar

disso, o futuro de toda a comunicação fixa de dados para distâncias superiores a alguns metros depende da fibra. Para obter mais informações sobre todos os aspectos das fibras ópticas e de suas redes, consulte (Hecht, 2001).

[T2] 2.3 Transmissão sem fios

Estamos assistindo ao surgimento de pessoas totalmente viciadas em informações: pessoas que precisam estar permanentemente on-line. Para esses usuários móveis, o par trançado, o cabo coaxial e a fibra óptica não têm a menor utilidade. Eles precisam transferir dados para seus computadores laptop, notebook, palmtop, de bolso ou de pulso sem depender da infra-estrutura de comunicação terrestre. A resposta para esses usuários está na comunicação sem fios. Nas próximas seções, examinaremos os conceitos básicos da comunicação sem fios em geral, pois ela tem muitas outras aplicações importantes além de oferecer conectividade aos usuários que desejam navegar na Web enquanto estão na praia.

Algumas pessoas chegam a acreditar que no futuro só haverá dois tipos de comunicação: as comunicações por fibra e as comunicações sem fios. Todos os computadores, telefones e equipamentos de fax fixos (isto é, não móveis) serão conectados por fibra óptica, e todos os computadores móveis utilizarão comunicações sem fios.

No entanto, existem algumas outras circunstâncias em que a comunicação sem fios apresenta vantagens até mesmo para dispositivos fixos. Por exemplo, quando há dificuldades para instalar cabos de fibra óptica em um prédio, devido a acidentes geográficos (montanhas, florestas, pântanos etc.), deve-se recorrer à tecnologia da transmissão sem fios. Não é à toa que a moderna comunicação digital sem fios teve início nas ilhas havaianas, onde os usuários estavam separados por grandes distâncias no oceano Pacífico e onde o sistema de

telefonia era totalmente inadequado.

[T3] 2.3.1 O espectro eletromagnético

Quando se movem, os elétrons criam ondas eletromagnéticas que podem se propagar pelo espaço livre (até mesmo no vácuo). Essas ondas foram previstas pelo físico inglês James Clerk Maxwell em 1865 e foram observadas pela primeira vez pelo físico alemão Heinrich Hertz em 1887. O número de oscilações por segundo de uma onda eletromagnética é chamado **freqüência**, f, e é medida em **Hz** (em homenagem a Heinrich Hertz). A distância entre dois pontos máximos (ou mínimos) consecutivos é chamada **comprimento de onda**, designado universalmente pela letra grega λ (lambda).

Quando se instala uma antena com o tamanho apropriado em um circuito elétrico, as ondas eletromagnéticas podem ser transmitidas e recebidas com eficiência por um receptor localizado a uma distância bastante razoável. Toda a comunicação sem fios é baseada nesse princípio.

No vácuo, todas as ondas eletromagnéticas viajam à mesma velocidade, independente de sua freqüência. Essa velocidade, geralmente chamada **velocidade da luz**, c, é aproximadamente igual a 3×10^8 m/s, ou cerca de 30 cm por nanossegundo. No cobre ou na fibra, a velocidade cai para cerca de 2/3 desse valor e se torna ligeiramente dependente da freqüência. A velocidade da luz é o limite máximo que se pode alcançar. Nenhum objeto ou sinal pode se mover com maior rapidez do que ela.

A relação fundamental entre f, λ e c (no vácuo) é:

$$\lambda f = c \tag{2-2}$$

Como c é uma constante, se conhecermos f, chegaremos a λ e vice-versa. Como uma regra prática, quando λ é medido em metros e f em MHz, [ver símbolo]. Por exemplo, ondas de 100 MHz têm cerca de 3 m de comprimento, ondas de 1000

MHz têm 0,3 metros, e ondas com 0,1 metro têm uma freqüência igual a 3000 MHz.

O espectro eletromagnético é mostrado na Figura 2.11. As porções de rádio, microondas, infravermelho e luz visível do espectro podem ser usadas na transmissão de informações, desde que seja modulada a amplitude, a freqüência ou a fase das ondas. A luz ultravioleta, os raios X e os raios gama representariam opções ainda melhores, por terem freqüências mais altas, mas são difíceis de produzir e modular, além de não se propagarem bem através dos prédios e serem perigosos para os seres vivos. As bandas (ou faixas) de freqüências listadas na parte inferior da Figura 2.11 são os nomes oficiais definidos pela ITU. Essas freqüências se baseiam nos comprimentos de onda; portanto, a banda LF vai de 1 a 10 km (aproximadamente, de 30 kHz a 300 kHz). Os termos LF, MF e HF são as abreviaturas, em inglês, de baixa, média e alta freqüência, respectivamente. É claro que, quando esses nomes foram criados, ninguém esperava ultrapassar 10 MHz, e assim foram atribuídos os seguintes nomes às bandas mais altas surgidas posteriormente: Very, Ultra, Super, Extremely e Tremendously High Frequency. Além desses não há outros nomes, mas Incredibly, Astonishingly e Prodigiously (IHF, AHF e PHF) ficariam muito bem.

[arte: ver original p. 101]

[Dísticos]

[1]f (Hz) 10^0 10^2 10^4 10^6 10^8 10^{10} 10^{12} 10^{14} 10^{16} 10^{18} 10^{20} 10^{22} 10^{24}

Rádio Microondas Infravermelho UV Raios X Raios gama

Luz visível

[2]f (Hz) 10^4 10^5 10^6 10^7 10^8 10^9 10^{10} 10^{11} 10^{12} 10^{13} 10^{14} 10^{15} 10^{16}

Par trançado Satélite Fibra óptica

Cabo coaxial Microonda terrestre

Marítimo Rádio AM Rádio FM

TV

[3]Banda LF MF HF VHF UHF SHF EHF THF

[F]Figura 2.11

[FL] O espectro eletromagnético e a maneira como ele é usado na comunicação

O volume de informações que uma onda eletromagnética é capaz de transportar está diretamente relacionado à sua largura de banda. Com a tecnologia atual, é possível codificar alguns bits por Hertz em freqüências baixas; no entanto, comumente esse número pode chegar a 8 em altas freqüências; assim, um cabo coaxial com uma largura de banda de 750 MHz pode transportar diversos gigabits/s. Observando a Figura 2.11, é possível entender com clareza por que as pessoas ligadas a redes têm um carinho todo especial pelas fibras ópticas.

Se resolvermos a Equação (2-2) para *f* e a diferenciarmos em relação a λ, obteremos:

[Inserir equação do O.A. p. 101]

Se decidirmos considerar as diferenças finitas em vez de diferenciais e trabalharmos apenas com valores absolutos, obteremos:

[Inserir equação do O.A. p. 102] (2-3)

Desse modo, com base na largura de uma banda de comprimentos de onda [ver símbolo], podemos calcular a banda de freqüências correspondente, [ver símbolo] e, a partir dela, a taxa de dados que a banda pode produzir. Quanto mais larga a banda, mais alta a taxa de dados. Como exemplo, considere a banda de 1,30 mícron da Figura 2.6. Nesse caso, temos $\lambda = 1{,}3 \times 10^{-6}$ e [ver símbolo] $= 0{,}17 \times 10^{-6}$; assim, [ver símbolo] é aproximadamente 30 THz. A 8 bits/Hz, por exemplo, teremos 240 Tbps.

A maioria das transmissões utiliza uma banda de freqüência estreita (ou seja, [ver símbolo]) para obter a melhor recepção (muitos watts/Hz). No entanto, em alguns casos, é usada uma banda larga, com duas variações. No **espectro de dispersão de salto de freqüência**, o transmissor salta de uma freqüência para outra centenas de vezes por segundo. Essa técnica é muito usada em comunicações militares, pois dificulta a detecção das transmissões e é praticamente impossível obstruí-las. Ela também oferece boa resistência ao esmaecimento de vários caminhos (multipath fading), porque o sinal direto sempre chega primeiro ao receptor. Os sinais refletidos percorrem caminhos mais longos e chegam depois. A essa altura, o receptor pode ter mudado de freqüência e não aceitar mais sinais na freqüência anterior, eliminando assim a interferência entre o sinal direto e os sinais refletidos. Há poucos anos, essa técnica também foi aplicada comercialmente — por exemplo, tanto as redes 802.11 quanto as Bluetooth a utilizam.

Como curiosidade, vale a pena mencionar que uma das pessoas que criaram essa técnica foi a atriz de cinema Hedy Lamarr, a deusa austríaca do sexo, a primeira mulher a aparecer nua em um filme cinematográfico (o filme tcheco de 1933 *Extase*). Seu primeiro marido era fabricante de armamentos e mostrou a ela como era fácil bloquear os sinais de rádio então empregados para controlar torpedos. Quando descobriu que ele estava vendendo armas a Hitler, ela ficou horrorizada, se disfarçou de criada para escapar dele e fugiu para Hollywood, a fim de continuar sua carreia como atriz de cinema. Em seu tempo livre, Hedy inventou o salto de freqüência para ajudar no esforço de guerra dos Aliados. Seu esquema utilizava 88 freqüências, o número de teclas (e freqüências) do piano. Por sua invenção, ela e seu amigo, o compositor George Antheil, receberam a patente 2.292.387 dos EUA. Porém, eles não conseguiram convencer a Marinha americana de que sua invenção tinha alguma utilidade prática, e nunca receberam royalties

por ela. Somente anos depois de expirar a patente, a invenção se tornou popular.

A outra a forma de espectro de dispersão, o **espectro de dispersão de seqüência direta**, que dispersa o sinal por uma ampla banda de freqüências, também está ganhando popularidade no mundo comercial. Em particular, alguns telefones celulares de segunda geração o empregam, e ele se tornará dominante com a terceira geração de telefonia móvel, graças à sua boa eficiência na utilização do espectro, à sua imunidade a ruídos e a outras propriedades. Algumas LANs sem fios também o utilizam. Voltaremos a estudar o espectro de dispersão mais adiante neste capítulo. Para obter informações mais detalhadas e fascinantes sobre a história da comunicação por espectro de dispersão, consulte (Scholtz, 1982).

Por enquanto, vamos partir da premissa de que todas as transmissões utilizam uma banda de freqüência estreita. A seguir, mostraremos como as diversas partes do espectro eletromagnético da Figura 2.11 são usadas, começando pelo rádio.

[T3] 2.3.2 Transmissão de rádio

As ondas de rádio são fáceis de gerar, podem percorrer longas distâncias e penetrar facilmente nos prédios; portanto, são amplamente utilizadas para comunicação, seja em ambientes fechados ou abertos. As ondas de rádio também são omnidirecionais, o que significa que elas viajam em todas as direções a partir da fonte; desse modo, o transmissor e o receptor não precisam estar cuidadosa e fisicamente alinhados.

Vale lembrar que o rádio omnidirecional nem sempre é bom. Na década de 1970, a General Motors decidiu equipar todos os seus novos Cadillacs com freios controlados por computador que impediam o travamento das rodas. Quando o motorista pisava no pedal de freio, o computador prendia e soltava os freios, em vez de travá-los de verdade. Um belo dia, um guarda rodoviário de Ohio começou

a usar seu novo rádio móvel para falar com a central de polícia e, de repente, o Cadillac próximo a ele passou a se comportar como um cavalo selvagem. Depois de ser abordado pelo patrulheiro, o motorista disse que não tinha feito nada e que o carro tinha ficado louco de uma hora para outra.

Eventualmente, começou a surgir um padrão: às vezes, os Cadillacs enlouqueciam, mas somente quando trafegavam pelas estradas de Ohio, particularmente quando estavam sendo observados por um guarda rodoviário. A General Motors demorou a entender o motivo pelo qual os Cadillacs funcionavam sem problemas nos outros estados e também em rodovias secundárias de Ohio. Só depois de muita pesquisa, eles descobriram que a fiação do Cadillac formava uma ótima antena que captava a freqüência usada pelo novo sistema de rádio da Patrulha Rodoviária de Ohio.

As propriedades das ondas de rádio dependem da freqüência. Em baixas freqüências, as ondas de rádio atravessam os obstáculos, mas a potência cai abruptamente à medida que a distância da fonte aumenta, cerca de $1/r^2$ no ar. Em altas freqüências, as ondas de rádio tendem a viajar em linha reta e a ricochetear nos obstáculos. Elas também são absorvidas pela chuva. Em todas as freqüências, as ondas de rádio estão sujeitas à interferência de motores e outros equipamentos elétricos.

Devido à capacidade que as ondas de rádio apresentam de percorrer longas distâncias, a interferência entre os usuários é um problema. Por essa razão, todos os governos exercem um rígido controle sobre o licenciamento do uso de transmissores de rádio, com apenas uma exceção, descrita a seguir.

Nas bandas VLF, LF e MF, as ondas de rádio se propagam perto do solo, como mostra a Figura 2.12(a). Essas ondas podem ser detectadas dentro de um raio de mil quilômetros nas freqüências mais baixas; porém, nas freqüências mais altas, esse raio de ação é bem menor. A radiodifusão em freqüências AM utiliza a banda

MF, razão pela qual as ondas de rádio produzidas pelas estações de rádio AM de Boston não podem ser captadas facilmente em Nova York. As ondas de rádio nessas bandas atravessam com facilidade os prédios; esse é o motivo por que os rádios portáteis funcionam em ambientes fechados. O principal problema relacionado à utilização dessas bandas em comunicação de dados diz respeito à baixa largura de banda que oferecem [ver Equação (2-2)].

Nas bandas HF e VHF, as ondas que se propagam ao longo do solo tendem a ser absorvidas pela terra. No entanto, as ondas que alcançam a ionosfera, uma camada de partículas carregadas situadas em torno da Terra a uma altura de 100 a 500 km, são refratadas por ela e enviadas de volta à Terra, como mostra a Figura 2.12(b). Em determinadas condições atmosféricas, os sinais podem ricochetear diversas vezes. Os operadores de radioamador utilizam essas bandas em comunicações de longa distância. Os militares também se comunicam nas bandas HF e VHF.

[arte: ver original p. 104]

[Dísticos]

[1] Ondas próximas ao solo

Superfície da Terra

(a)

[2] Ionosfera

Superfície da Terra

(b)

[F]Figura 2.12

[FL] (a) Nas bandas VLF, VF e MF, as ondas de rádio obedecem à curvatura da Terra. (b) Na banda HF, elas ricocheteiam na ionosfera

[T3] 2.3.3 Transmissão de microondas

Acima de 100 MHz, as ondas trafegam praticamente em linha reta e, portanto, podem ser concentradas em uma faixa estreita. A concentração de toda a energia em um pequeno feixe através de uma antena parabólica (como a conhecida antena de TV por satélite) oferece uma relação sinal/ruído muito mais alta, mas as antenas de transmissão e recepção devem estar alinhadas com o máximo de precisão. Além disso, essa direcionalidade permite o alinhamento de vários transmissores em uma única fileira, fazendo com que eles se comuniquem com vários receptores também alinhados sem que haja interferência, desde que sejam observadas algumas regras mínimas de espaçamento. Antes da fibra óptica, durante décadas essas microondas formaram o núcleo do sistema de transmissão telefônica de longa distância. Na verdade, a MCI, uma das primeiras concorrentes da AT&T após sua desregulamentação, construiu todo o seu sistema com comunicações de microondas que percorriam dezenas de quilômetros entre uma torre e outra. Até mesmo o nome a empresa refletia isso (MCI significava Microwave Communications, Inc.). Há muito tempo, a MCI passou a utilizar as fibras ópticas e se fundiu à WorldCom.

Tendo em vista que as microondas viajam em linha reta, se as torres estiverem muito afastadas, a Terra acabará ficando entre elas (como acontece no caso de um enlace entre San Francisco e Amsterdam). Conseqüentemente, é preciso instalar repetidores a intervalos periódicos. Quanto mais altas são as torres, mais distantes elas podem estar umas da outras. A distância entre os repetidores aumenta de acordo com a raiz quadrada da altura da torre. Torres com 100 m de altura devem ter repetidores a cada 80 km.

Ao contrário das ondas de rádio nas freqüências mais baixas, as microondas não atravessam muito bem as paredes dos edifícios. Além disso, muito embora o feixe possa estar bem concentrado no transmissor, ainda há alguma divergência no espaço. Algumas ondas podem ser refratadas nas camadas atmosféricas mais

baixas e, conseqüentemente, sua chegada pode ser mais demorada que a das ondas diretas. As ondas retardadas podem chegar fora de fase em relação à onda direta, e assim cancelar o sinal. Esse efeito é chamado **esmaecimento de vários caminhos** (multipath fading) e costuma provocar sérios problemas. Ele depende das condições atmosféricas e da freqüência. Algumas operadoras mantêm 10% dos seus canais ociosos como sobressalentes; esses canais serão utilizados quando o esmaecimento de vários caminhos eliminar temporariamente alguma banda de freqüência.

A demanda por mais e mais espectro serve para manter o processo de aperfeiçoamento tecnológico, permitindo que as transmissões utilizem freqüências cada vez mais altas. As bandas de até 10 GHz agora são de uso rotineiro, mas a partir de 4 GHz surge um novo problema: a absorção pela água. Essas ondas têm apenas alguns centímetros e são absorvidas pela chuva. Esse efeito não causaria problema algum se estivéssemos planejando construir um gigantesco forno de microondas para ser usado a céu aberto mas, no caso das comunicações, trata-se de um grave problema. Assim como acontece com o esmaecimento de vários caminhos, a única solução é desligar os enlaces que estão sendo afetados pela chuva e criar uma nova rota que os contorne.

Em resumo, a comunicação por microondas é muito usada na telefonia à longa distância, em telefones celulares, na distribuição de sinais de televisão e em outros usos que uma severa diminuição do espectro obrigou a desenvolver. Ela têm uma série de vantagens significativas sobre a fibra. A mais importante delas é que as microondas dispensam a necessidade de se ter direitos sobre um caminho. Além do mais, quando se compra um pequeno lote de terra a cada 50 quilômetros e nele é instalada uma torre de microondas, é possível ignorar o sistema telefônico e se comunicar diretamente. Foi por essa razão que a MCI mudou de orientação com tanta rapidez, tornando-se uma companhia telefônica

de longa distância. (A Sprint trilhou um caminho bem diferente: ela se formou a partir da Southern Pacific Railroad, que já detinha um grande número de concessões de direitos de percurso, e simplesmente enterrava os cabos de fibra ao lado das ferrovias.)

O uso de microondas também é relativamente econômico. A instalação de duas torres simples (com alguns postes com quatro esteios) e a colocação de antenas em cada uma delas pode ser menos dispendiosa que enterrar 50 quilômetros de fibra em uma área urbana congestionada ou em uma região montanhosa, e talvez seja mais econômica que arrendar a rede de fibra da companhia telefônica, especialmente se esta ainda não tiver coberto totalmente os custos da retirada do cobre quando os cabos de fibra foram instalados.

[T4] A política do espectro eletromagnético

Para evitar o caos total, têm sido feitos acordos nacionais e internacionais a respeito de quem terá o direito de usar cada uma das freqüências. Como todos querem uma taxa de dados mais alta, todos desejam um espectro maior. Os governos nacionais alocam bandas do espectro para rádio AM e FM, televisão e telefones celulares, como também para as empresas de telefonia, a polícia, os usuários marítimos, de navegação, militares, do governo, e para muitos outros usuários concorrentes. Em termos mundiais, uma agência da ITU-R (WARC) tenta coordenar essa alocação de forma que possam ser fabricados dispositivos que funcionem em vários países. Porém, os países não são limitados pelas recomendações da ITU-R, e a FCC (Federal Communications Commission), que realiza a alocação para os Estados Unidos, ocasionalmente tem rejeitado recomendações da ITU-R (em geral, porque elas exigiam que algum grupo politicamente poderoso desistisse de alguma fração do espectro).

Até mesmo quando uma parte do espectro é alocada para algum uso, como

telefones celulares, existe a questão adicional de decidir qual concessionária terá permissão para usar cada uma das freqüências. Três algoritmos foram extensamente usados no passado. O algoritmo mais antigo, freqüentemente chamado **concurso de beleza**, exige que cada concessionária explique por que sua proposta serve melhor ao interesse público. Os funcionários do governo decidem então qual dentre as belas histórias mais lhes agrada. Fazer algum funcionário do governo oferecer como prêmio a propriedade de bilhões de dólares à sua empresa favorita em geral leva a suborno, corrupção, nepotismo e crimes piores. Além disso, até mesmo um funcionário do governo honesto e escrupuloso que imagine que uma companhia estrangeira poderia realizar um trabalho melhor que qualquer das empresas nacionais teria muito a explicar. Essa observação levou ao algoritmo 2, que realiza um sorteio entre as empresas interessadas. O problema com essa idéia é que empresas que não têm nenhum interesse em usar o espectro podem participar do sorteio. Se, digamos, um restaurante de fast-food ou uma cadeia de sapatarias ganhasse, a empresa poderia revender o espectro a uma concessionária com um enorme lucro e sem qualquer risco.

A idéia de conceder fatias do espectro a empresas com uma enorme dose de sorte mas sem qualquer método tem sido severamente criticada por muitos, o que levou ao algoritmo 3: realizar leilões e conceder a largura de banda à empresa que fizer a melhor proposta. Quando a Inglaterra leiloou as freqüências necessárias para os sistemas de telefonia móvel em 2000, o governo esperava obter aproximadamente 4 bilhões de dólares. Na realidade, recebeu quase 40 bilhões de dólares, porque as concessionárias entraram em uma disputa frenética, mortas de medo de perderem o barco da telefonia móvel. Esse evento despertou a ganância dos governos vizinhos e os inspirou a realizar seus próprios leilões. Isso funcionou, mas também deixou algumas concessionárias tão

endividadas que elas chegaram perto da bancarrota. Até mesmo nos melhores casos, irá demorar muitos anos para essas empresas recuperarem o custo do licenciamento.

Uma abordagem muito diferente para alocar freqüências é simplesmente não alocá-las. Basta deixar todo mundo transmitir à vontade, mas regular a potência utilizada, de forma que as estações tenham um alcance tão pequeno que não possam interferir umas com as outras. De acordo com essa proposta, a maioria dos governos reserva algumas bandas de freqüência, chamadas bandas **ISM (Industrial, Scientific, Medical)** para uso sem licença. Sistemas para abertura de portas de garagens, telefones sem fios, brinquedos controlados por rádio, dispositivos de indicação sem fio e vários outros aparelhos domésticos sem fios utilizam as bandas ISM. Para minimizar a interferência entre esses dispositivos não coordenados, a FCC estabelece que todos os dispositivos nas bandas ISM devem utilizar técnicas de espectro de dispersão. Regras semelhantes se aplicam em outros países.

A localização das bandas ISM varia de país para país. Por exemplo, nos Estados Unidos, dispositivos cuja potência está abaixo de 1 watt podem usar as bandas mostradas na Figura 2.13 sem exigir uma licença da FCC. A banda de 900 MHz funciona melhor, mas está lotada e não se encontra disponível em todo o mundo. A banda de 2,4 GHz está disponível na maioria dos países, mas é sujeita a interferências de fornos de microondas e instalações de radar. A rede Bluetooth e algumas LANs sem fios que seguem o padrão 802.11 operam nessa banda. A banda de 5,7 GHz é nova e relativamente pouco desenvolvida, e assim o equipamento para ela é dispendioso; porém, à medida que as redes 802.11a a utilizarem, ela logo se tornará mais popular.

[T3] 2.3.4 Ondas de infravermelho e milimétricas

As ondas de infravermelho e ondas milimétricas sem guias são extensamente utilizadas na comunicação de curto alcance. Todos os dispositivos de controle remoto utilizados nos aparelhos de televisão, videocassetes e equipamentos estereofônicos empregam a comunicação por infravermelho. Eles são relativamente direcionais, econômicos e fáceis de montar, mas têm uma desvantagem importante: não atravessam objetos sólidos (para provar essa tese, posicione-se entre o controle remoto e o televisor). Em geral, quando nos deslocamos do rádio de onda longa em direção à luz visível, as ondas assumem um comportamento cada vez mais parecido com o da luz, perdendo pouco a pouco as características de ondas de rádio.

[arte: imagem original da p. 107]

[Dísticos]

[1] Largura de banda 26 MHz 83,5 MHz 125 MHz

[2] Freqüência 902 MHz 928 MHz 2,4 GHz 2,4835 GHz 5,734 GHz 5,860 GHz

[F]Figura 2.13

[FL] As bandas ISM nos Estados Unidos

Por outro lado, o fato de as ondas de infravermelho não atravessarem paredes sólidas pode ser visto como uma qualidade. É por essa razão que um sistema infravermelho instalado em um ambiente fechado não interfere em um sistema semelhante instalado nas salas ou nos prédios adjacentes: não é possível controlar o aparelho de televisão do vizinho com o seu controle remoto. Além disso, a segurança dos sistemas de infravermelho contra espionagem é melhor que a dos sistemas de rádio, exatamente por essa razão. Portanto, não é necessária nenhuma licença do governo para operar um sistema de infravermelho, ao contrário dos sistemas de rádio, que devem ser licenciados fora

das bandas ISM. A comunicação por infravermelho tem uso limitado em escritórios, por exemplo, para conectar notebooks e impressoras, mas não deverá ter um papel importante no jogo das comunicações.

[T3] 2.3.5 Transmissão por ondas de luz

A sinalização óptica sem guia vem sendo utilizada há séculos. Uma aplicação mais moderna consiste em conectar as LANs em dois prédios por meio de lasers instalados em seus telhados. Por sua própria natureza, a sinalização óptica coerente que utiliza raios laser é unidirecional; assim, cada prédio precisa do seu próprio raio laser e do seu próprio fotodetector. Esse esquema oferece uma largura de banda muito alta a um custo bastante baixo. Ele também é relativamente fácil de ser instalado e, ao contrário das microondas, não precisa de uma licença da FCC.

Nesse caso, a principal virtude do laser, um feixe muito estreito, também é sua fraqueza. Apontar um feixe de raios laser com 1 mm de largura para um alvo com o tamanho da cabeça de um alfinete a 500 metros de distância exige uma pontaria quase impossível. Em geral, são colocadas lentes no sistema para desfocar levemente o feixe.

Uma das desvantagens dos feixes de raios laser é o fato de que eles não podem atravessar chuva ou neblina espessa, mas normalmente funcionam bem em dias ensolarados. Contudo, o autor participou certa vez de uma conferência em um moderno hotel europeu cujos organizadores tiveram a felicidade de oferecer uma sala repleta de terminais para que os participantes pudessem ler suas mensagens de correio eletrônico durante as apresentações menos interessantes. Como o PTT local não se dispôs a instalar um grande número de linhas telefônicas que após três dias seriam desativadas, os organizadores colocaram um raio laser no telhado e o apontaram na direção do prédio de ciência da computação da

universidade onde trabalhavam, situada há alguns quilômetros dali. Eles testaram o sistema na noite anterior à conferência e ele funcionou perfeitamente. Às 9h da manhã seguinte, em um belo dia de sol, o sistema entrou em pane e ficou fora do ar durante todo o dia. Naquela noite, os organizadores voltaram a testá-lo com todo o cuidado e, mais uma vez, tudo funcionou de forma absolutamente perfeita. Nos dois dias seguintes, o problema se repetiu.

Após a conferência, os organizadores descobriram o problema. O calor do sol fez com que emanassem correntes de convecção do telhado do prédio, como mostra a Figura 2.14. Esse ar turbulento desviou o feixe e fez com que ele dançasse em torno do detector. Esse tipo de "visão" atmosférica faz as estrelas cintilarem (e é por essa razão que os astrônomos instalam seus telescópios nos cumes das montanhas — para ficarem acima do maior volume possível de atmosfera). Esse mesmo ar também é responsável pelas estradas bruxuleantes em dias quentes e pelas imagens tremidas quando olhamos para fora do automóvel sobre um radiador quente.

[arte: ver original p. 108]

[Dísticos]

[1] Fotodetector

[2] Região de visão turbulenta

[3] Ondas de calor elevando-se do edifício

[4] O feixe de raios laser não atinge o detector

[5] Laser

[F]Figura 2.14

[FL] Correntes de convecção podem interferir nos sistemas de comunicação a laser. A figura mostra um sistema bidirecional com dois lasers

==

[T2] 2.4 Satélites de comunicações

Na década de 1950 e no início dos anos 60, as pessoas tentavam configurar sistemas de comunicações emitindo sinais que se refletiam em balões meteorológicos metalizados. Infelizmente, os sinais recebidos eram muito fracos para que tivessem algum uso prático. Em seguida, a Marinha dos Estados Unidos detectou uma espécie de balão meteorológico que ficava permanentemente no céu — a Lua — e criou um sistema operacional para comunicações entre o navio e a base que utilizava a Lua em suas transmissões.

O progresso no campo da comunicação celeste precisou esperar até que o primeiro satélite de comunicações fosse lançado. A principal diferença entre um satélite artificial e um real é que o artificial amplifica os sinais antes de enviá-los de volta, transformando uma estranha curiosidade em um avançado sistema de comunicações.

Os satélites de comunicações possuem algumas propriedades interessantes, que os tornam atraentes para muitas aplicações. Em sua forma mais simples, um satélite de comunicações pode ser considerado um grande repetidor de microondas no céu. Ele contém diversos **transponders**; cada um deles ouve uma parte do espectro, amplifica os sinais de entrada e os transmite novamente em outra freqüência, para evitar interferência com o sinal de entrada. Os feixes descendentes podem ser largos, cobrindo uma fração substancial da superfície terrestre, ou estreitos, cobrindo uma área com apenas centenas de quilômetros de diâmetro. Esse modo de operação é conhecido como **canal em curva (bent pipe)**.

De acordo com a lei de Kepler, o período orbital de um satélite varia de acordo com o raio da órbita elevado à potência 3/2. Quanto mais alto o satélite, mais longo o período. Perto da superfície da Terra, o período é de cerca de 90

minutos. Conseqüentemente, os satélites de baixa órbita saem de visão com bastante rapidez, e assim são necessários muitos deles para proporcionar cobertura contínua. A uma altitude de aproximadamente 35.800 km, o período é de 24 horas. Na altitude de 384.000 km, o período é de cerca de um mês, como pode atestar qualquer pessoa que observe a Lua regularmente.

O período do satélite é importante, mas não é o único fator para se determinar onde posicioná-lo. Outra questão é a presença dos cinturões de Van Allen, camadas de partículas altamente carregadas que são capturadas pelo campo magnético terrestre. Qualquer satélite em órbita dentro deles seria destruído com bastante rapidez pelas partículas carregadas com alta energia presas pelo campo magnético da Terra. Esses fatores nos levam a identificar três regiões nas quais os satélites podem ser posicionados com segurança. Essas regiões e algumas de suas propriedades estão ilustradas na Figura 2.15. Descreveremos rapidamente a seguir os satélites que habitam cada uma dessas regiões.

[T3] 2.4.1 Satélites geoestacionários

Em 1945, o escritor de ficção científica Arthur C. Clarke calculou que um satélite na altitude de 35.800 km em órbita circular equatorial pareceria permanecer imóvel no céu, e assim não precisaria ser rastreado (Clarke, 1945). Ele continuou a descrever um sistema de comunicação completa que usava esses **satélites geoestacionários** ou **satélites geossíncronos** (tripulados), incluindo as órbitas, os painéis solares, as freqüências de rádio e os procedimentos de lançamento. Infelizmente, ele concluiu que os satélites eram impraticáveis devido à impossibilidade de colocar em órbita amplificadores a válvulas, frágeis e ávidos por energia; assim, nunca levou sua idéia adiante, embora tenha escrito algumas histórias de ficção científica sobre ela.

[arte: ver pág. original 110]

[Tabela]

Altitude (km)	Tipo	Latência (ms)	Satélites necessários
35.000	GEO	270	3
30.000			
25.000			
20.000	Cinturão de Van Allen superior		
15.000			
10.000	MEO	35-85	10
5.000	Cinturão de Van Allen inferior		
0		1-7	50
	LEO		

[F]Figura 2.15

[FL] Satélites de comunicações e algumas de suas propriedades, inclusive altitude acima da Terra, tempo de retardo de ida e volta, e ainda o número de satélites necessários para cobertura global

A invenção do transistor mudou tudo isso, e o primeiro satélite artificial de comunicações, chamado Telstar, foi lançado em julho de 1962. Desde então, os satélites de comunicações se transformaram em um negócio de vários bilhões de dólares, e o único aspecto do espaço sideral que se tornou altamente lucrativo. Esses satélites de alta órbita são chamados com freqüência satélites **GEO (Geoestationary Earth Orbit).**

Com a tecnologia atual, não é muito inteligente ter satélites geoestacionários com espaçamento muito menor que 2 graus entre eles no plano equatorial de 360 graus, a fim de evitar interferência. Com um espaçamento de 2 graus, só pode haver 360/2 = 180 desses satélites no céu ao mesmo tempo. No entanto, cada transponder pode usar várias freqüências e polarizações, com a finalidade de

aumentar a largura de banda disponível.

Para evitar o caos total no céu, a alocação de slots de órbitas é feita pela ITU. Esse processo é altamente político, com países que mal saíram da idade da pedra exigindo "seus" slots de órbitas (com a finalidade de arrendá-los pela melhor oferta). Contudo, outros países sustentam que os direitos nacionais de propriedade não estendem para cima até a Lua e que nenhum país tem direito legal sobre os slots de órbita acima de seu território. Para aumentar a disputa, as telecomunicações comerciais não são a única aplicação. Emissoras de televisão, governos e instituições militares também querem ter uma fatia da grande torta orbital.

Os satélites modernos podem ser bastante grandes, pesando até 4000 kg e consumindo vários quilowatts de energia elétrica produzida pelos painéis solares. Os efeitos da gravidade solar, lunar e planetária tendem a movê-los para fora de seus slots de órbita e de suas orientações, um efeito compensado por motores de foguetes a bordo. Essa atividade de ajuste fino é chamada **manutenção da estação.** Porém, quando o combustível para os motores tiver se esgotado, em geral no período de 10 anos, o satélite fica sem controle, e portanto tem de ser desativado. Eventualmente, a órbita decai e o satélite entra de novo na atmosfera e é queimado ou, às vezes, colide com a Terra.

Os slots de órbita não são o único ponto de discórdia. As freqüências também são, porque as transmissões do satélite para a Terra (downlink) interferem com usuários de microondas. Conseqüentemente, a ITU alocou certas bandas de freqüência para usuários de satélites. As principais estão listadas na Figura 2.16. A banda C foi a primeira a ser designada para tráfego comercial de satélite. Duas faixas de freqüências são atribuídas nessa banda, a inferior para tráfego downlink (do satélite) e a superior para tráfego uplink (para o satélite). Para permitir que o tráfego ocorra em ambos os sentidos ao mesmo tempo, são necessários dois

canais, um para cada sentido. Essas bandas já estão sobrecarregadas, porque também são usadas pelas concessionárias de telecomunicações nos enlaces terrestres de microondas. As bandas L e S foram acrescentadas por um acordo internacional de 2000. Porém, elas são estreitas e estão lotadas.

[arte: ver original p. 111]

[T]Tabela

Banda	Downlink	Uplink	Largura de banda	Problemas
L	1,5 GHz	1,6 GHz	15 MHz	Baixa largura de banda; lotada
S	1,9 GHz	2,2 GHz	70 MHz	Baixa largura de banda; lotada
C	4,0 GHz	6,0 GHz	500 MHz	Interferência terrestre
Ku	11 GHz	14 GHz	500 MHz	Chuva
Ka	20 GHz	30 GHz	3500 MHz	Chuva; custo do equipamento

[F]Figura 2.16

[FL] As principais bandas de satélite

A próxima banda mais alta disponível para concessionárias de telecomunicações comerciais é a banda Ku (K under). Essa banda (ainda) não está congestionada e, nessas freqüências, os satélites podem ficar à distância de apenas 1 grau. Entretanto, existe um outro problema: a chuva. A água é um grande absorvente dessas microondas curtas. Felizmente, em geral as tempestades fortes costumam ser localizadas; assim, o uso de várias estações terrestres separadas por uma grande distância, em lugar de apenas uma, contorna o problema, mas ao preço de antenas, cabos e equipamentos eletrônicos extras para permitir a comutação rápida entre estações. Na banda Ka (K above) também foi alocada largura de banda para o tráfego de satélite comercial, mas o equipamento necessário para usá-la ainda continua caro. Além dessas bandas comerciais, também existem muitas bandas governamentais e militares.

Um satélite moderno tem cerca de 40 transponders, cada um com uma largura de banda de 80 MHz. Em geral, cada transponder opera como um canal em curva, mas satélites recentes têm alguma capacidade de processamento a bordo, permitindo operação mais sofisticada. Nos primeiros satélites, a divisão dos transponders em canais era estática: a largura de banda era simplesmente dividida em bandas de freqüências fixas. Hoje em dia, o feixe de cada transponder é divido em slots de tempo, com diversos usuários realizando turnos de atividades. Estudaremmos essas duas técnicas (a multiplexação por divisão de freqüência e a multiplexação por divisão de tempo) em detalhes mais adiante neste capítulo.

Os primeiros satélites geoestacionários tinham um único feixe espacial que iluminava cerca de 1/3 da superfície da Terra, denominado sua área de cobertura (footprint). Com o enorme declínio de preço, tamanho e requisitos de potência dos equipamentos microeletrônicos, tornou-se viável uma estratégia de transmissão muito mais sofisticada. Cada satélite é equipado com diversas antenas e vários transponders. Cada feixe descendente pode ser focalizado em uma pequena área geográfica; portanto, podem acontecer diversas transmissões ascendentes e descendentes ao mesmo tempo. Em geral, esses **feixes pontuais** têm forma elíptica e podem ter apenas algumas centenas de quilômetros de diâmetro. Em geral, um satélite de comunicações para os Estados Unidos tem um único feixe para os 48 estados contíguos, além de feixes pontuais para o Alasca e o Havaí.

Um novo desenvolvimento no mundo dos satélites de comunicações é a criação de microestações de baixo custo, às vezes chamadas **VSATs (Very Small Aperture Terminals)** (Abramson, 2000). Esses pequenos terminais têm antenas de 1 metro ou menores (em comparação com 10 metros para uma antena de GEO padrão) e podem emitir cerca de 1 watt de energia. Geralmente, o uplink é adequado para

19,2 kbps, mas o downlink com freqüência exige 512 kbps ou mais. A televisão de difusão direta por satélite utiliza essa tecnologia na transmissão de mão única. Em muitos sistemas VSAT, as microestações não têm energia suficiente para se comunicarem diretamente com as outras (via satélite, é óbvio). Em vez disso, é necessária uma estação terrestre especial, o **hub**, com uma grande antena de alto ganho para retransmitir o tráfego entre VSATs, como mostra a Figura 2.17. Nesse modo de operação, o transmissor ou o receptor possui uma grande antena e um amplificador de grande potência. O compromisso é um retardo mais longo em troca de estações mais econômicas para o usuário final.

Os VSATs apresentam um grande potencial em áreas rurais. Ele não é amplamente apreciado; porém mais de metade da população do mundo vive a uma distância de no máximo uma hora a pé do telefone mais próximo. Estender fios telefônicos até milhares de pequenas aldeias é algo que vai muito além do orçamento da maioria dos governos do terceiro mundo, mas a instalação de antenas VSAT de 1 metro de diâmetro alimentadas por células solares com freqüência é algo viável. Os VSATs fornecem a tecnologia que irá conectar o mundo.

Os satélites de comunicações têm diversas propriedades radicalmente diferentes dos enlaces terrestres ponto a ponto. Para começar, embora os sinais enviados e recebidos por um satélite trafeguem à velocidade da luz (aproximadamente 300.000 km/s), a longa distância de ida e volta introduz um retardo substancial para os satélites GEO. Dependendo da distância entre o usuário e a estação terrestre, e também da elevação do satélite acima do horizonte, o tempo total de trânsito está entre 250 e 300 ms. Um valor típico é 270 ms (540 ms, no caso de um sistema VSAT com um hub).

Para fins de comparação, os enlaces de microondas terrestres têm um retardo de propagação de aproximadamente 3 μs/km, e os enlaces de cabo coaxial ou fibra

óptica geram um retardo de cerca de 5 µs/km. Nesse último caso, o retardo é maior, porque os sinais eletromagnéticos trafegam com maior rapidez no ar que em materiais sólidos.

[arte: ver original p. 113]

[Dísticos]

[1]Satélite de comunicações

[2]Hub

[F]Figura 2.17

[FL] VSATs utilizando um hub

Outra propriedade importante dos satélites é que eles são basicamente meios de difusão. Enviar uma mensagem para milhares de estações localizadas na área de cobertura de um transponder não custa mais do que enviar a mensagem para apenas uma estação. Para algumas aplicações, essa propriedade é muito útil. Por exemplo, poderíamos imaginar um satélite transmitindo páginas a Web populares para os caches de um grande número de computadores espalhados por uma área extensa. Mesmo quando a difusão pode ser simulada com o uso de linhas ponto a ponto, a difusão por satélite pode ser muito mais econômica. Por outro lado, do ponto de vista da segurança e da privacidade, os satélites são um completo desastre: todo mundo pode ouvir tudo. A criptografia é essencial quando a segurança é necessária.

Nos satélites, o custo de transmissão de uma mensagem é independente da distância percorrida. O serviço de uma chamada transcontinental não custa mais do que uma chamada entre um lado e outro da rua. Os satélites também proporcionam excelentes taxas de erros e podem ser explorados quase instantaneamente, um detalhe fundamental para a comunicação militar.

[T3] 2.4.2 Satélites terrestres de órbita média

Em altitudes muito mais baixas, entre os dois cinturões de Van Allen, encontramos os satélites **MEO (Medium-Earth Orbit).** Vistos da Terra, esses satélites se deslocam lentamente em longitude, levando cerca de 6 horas para circular a Terra. Conseqüentemente, eles devem ser acompanhados à medida que se movem pelo céu. Pelo fato de estarem em órbitas mais baixas que os GEOs, eles têm uma área de cobertura menor no solo e exigem transmissores menos potentes para alcançá-los. Atualmente, esses satélites não são usados para telecomunicações, e assim não os examinaremos mais aqui. Os 24 satélites **GPS (Global Positioning System)** que estão em órbita a cerca de 18.000 km de altitude são exemplos de satélites MEO.

[T3] 2.4.3 Satélites terrestres de baixa órbita

A uma altitude menor, encontramos os satélites **LEO (Low-Earth Orbit).** Devido a seu rápido movimento, são necessárias grandes quantidades desses satélites para formar um sistema completo. Por outro lado, pelos fato de os satélites estarem muito próximos da Terra, as estações terrestres não precisam de muita potência, e o retardo de ida e volta é de apenas alguns milissegundos. Nesta seção, examinaremos três exemplos, dois deles destinados às comunicações de voz e um destinado ao serviço da Internet.

[T4] Iridium

Conforme mencionamos antes, durante os primeiros 30 anos da era do satélite, os satélites de baixa órbita raramente eram usados para comunicação, porque apareciam e desapareciam de vista com muita rapidez. Em 1990, a Motorola deu início a um novo empreendimento e enviou um requerimento à FCC, solicitando permissão para lançar 77 satélites de baixa órbita do projeto Iridium (o elemento

77 é o irídio). Mais tarde, o plano foi revisto no sentido de se usar apenas 66 satélites; assim, o projeto deveria ter seu nome alterado para Dysprosium (o elemento 66 é o disprósio), mas esse nome provavelmente lembrava muito mais uma doença do que um satélite. A idéia era que assim que um satélite estivesse fora de vista, outro o substituiria. Essa proposta criou uma agitação entre outras empresas de comunicações. De repente, todas elas quiseram lançar uma cadeia de satélites de baixa órbita.

Após sete anos reunindo parceiros e financiamentos, os parceiros lançaram os satélites Iridium em 1997. O serviço de comunicação se iniciou em novembro de 1998. Infelizmente, a demanda comercial por grandes e pesados telefones via satélite era desprezível, porque a rede de telefonia móvel (celular) havia crescida de modo espetacular desde 1990. Como conseqüência, O Iridium não gerou lucro e foi à bancarrota em agosto de 1999, em um dos mais espetaculares fiascos corporativos da história. Os satélites e outros bens (no valor de 5 bilhões de dólares) foram adquiridos mais tarde por um investidor por 25 milhões de dólares, em uma espécie de venda de garagem extraterrestre. O serviço Iridium foi reiniciado em março de 2001.

O objetivo básico do Iridium era (e ainda é) fornecer um serviço de telecomunicações de amplitude mundial por meio de dispositivos portáteis que se comunicam diretamente com os satélites Iridium. Há serviços de voz, dados, busca, fax e navegação em qualquer lugar do mundo, seja em terra, mar e ar. Os clientes incluem as indústrias marítima, de aviação e de exploração de petróleo, bem como pessoas que viajam para regiões do mundo que não têm uma infra-estrutura de telecomunicações (por exemplo, desertos, montanhas, selvas e alguns países do terceiro mundo).

Os satélites Iridium estão posicionados a uma altitude de 750 km, em órbitas polares circulares. Eles estão organizados em eixos norte-sul, com um satélite a

cada 32 graus de latitude. Com seis eixos de satélites, toda a Terra é coberta, como sugere a Figura 2.18(a). As pessoas com poucos conhecimentos de química podem pensar nessa disposição como um imenso átomo de disprósio, tendo a Terra como o núcleo e os satélites como elétrons.

[arte: ver original p. 115]

[Dísticos]

[1] (a) (b)

[F]Figura 2.18

[FL] (a) Os satélites Iridium formam seis eixos em torno da Terra. (b) 1628 células móveis cobrem a Terra

Cada satélite tem no máximo 48 células (feixes pontuais), com um total de 1628 células sobre a superfície da Terra, como mostra a Figura 2.18(b). Cada satélite tem a capacidade de 3840 canais, ou 253.440 canais ao todo. Alguns deles são usados para busca e navegação, enquanto outros são empregados para dados e voz.

Uma propriedade interessante do Iridium é que a comunicação entre clientes distantes ocorre no espaço, com um satélite retransmitindo dados para o seguinte, como ilustra a Figura 3.19(a). Na figura, vemos um chamador no Pólo Norte entrando em contato com um satélite situado diretamente acima dele. A chamada é retransmitida por outros satélites e, finalmente, chega ao destino no Pólo Sul.

[T4] Globalstar

Um projeto alternativo para o Iridium é o Globalstar. Ele se baseia em 48 satélites LEO, mas utiliza um esquema de comutação diferente do que é usado no Iridium. Enquanto o Iridium retransmite as chamadas de satélite para satélite, o que exige

sofisticado equipamento de comutação nos satélites, o Globalstar utiliza um projeto tradicional de canal em curva. A chamada originada no Pólo Norte na Figura 2.19(b) é enviado de volta a Terra e recebida pela grande estação terrestre em Santa's Workshop. A chamada é então roteada por uma rede terrestre até a estação terrestre mais próxima ao destino, e é entregue por uma conexão de canal em curva da maneira ilustrada. A vantagem desse esquema é que ele coloca a maior parte da complexidade no solo, onde ela é mais fácil de administrar. Além disso, o uso de grandes antenas nas estações terrestres, capazes de emitir um sinal potente e receber um sinal fraco significa que podem ser utilizados telefones de potência mais baixa. Afinal, o telefone emite apenas alguns miliwatts de potência, e assim o sinal que volta para a estação terrestre é bastante fraco, mesmo depois de ter sido amplificado pelo satélite.

[arte: ver pág. original 116]

[Dísticos]

[1]Switches de satélites no espaço

(a)

[2]Satélite de canal em curva

Comutação no solo

(b)

[F]Figura 2.19

[FL] (a) Retransmissão do espaço. (b) Retransmissão no solo

[T4] Teledesic

O Iridium se destina a usuários de telefonia localizados em lugares estranhos. Nosso próximo exemplo, o **Teledesic**, se destina a usuários da Internet ávidos por largura de banda, existentes em todo o mundo. Ele foi concebido em 1990 pelo pioneiro da telefonia móvel (celular) Craig McCaw e pelo fundador da Microsoft

Bill Gates, que não estava satisfeito com o ritmo lento com que as empresas de telefonia de todo o mundo estavam fornecendo alta largura de banda a usuários de computadores. O objetivo do sistema Teledesic é fornecer a milhões de usuários da Internet concorrentes um uplink de até 100 Mbps e um downlink de até 720 Mbps, usando uma pequena antena fixa do tipo VSAT, ignorando por completo o sistema de telefonia. Para as empresas de telefonia, isso é uma torta no céu.

O projeto original era um sistema que consistia em 288 satélites de área de cobertura pequena, organizados em 12 planos imediatamente abaixo do cinturão de Van Allen inferior, a uma altitude igual a 1350 km. Mais tarde, essa organização mudou para 30 satélites com área de cobertura maior. A transmissão ocorre na banda Ka, relativamente pouco ocupada e com alta largura de banda. O sistema é comutado por pacotes no espaço, com cada satélite capaz de rotear pacotes até os satélites vizinhos. Quando um usuário necessita de largura de banda para enviar pacotes, ela é solicitada e atribuída dinamicamente em cerca de 50 ms. O sistema está programado para funcionar em 2005, se tudo correr como planejado.

[T3] 2.4.4 Comparação entre satélites e fibra óptica

Uma comparação entre a comunicação por satélite e a comunicação terrestre é instrutiva. Há 20 anos, pensava-se que o futuro da comunicação residia nos satélites de comunicações. Afinal de contas, o sistema telefônico mudou muito pouco nos últimos 100 anos e não mostrou sinais de mudança para os próximos 100 anos. Esse movimento glacial foi causado em grande parte pelo ambiente regulador no qual se esperava que as companhias telefônicas fornecessem bons serviços de voz a preços razoáveis (o que elas fizeram) e, em troca, tivessem lucro garantido sobre seu investimento. Havia modems de 1200 bps disponíveis

para as pessoas que precisavam transmitir dados. Isso era praticamente tudo o que existia na época.

Com o surgimento da concorrência, em 1984 nos Estados Unidos e um pouco mais tarde na Europa, esse quadro se alterou radicalmente. As companhias telefônicas começaram a substituir suas redes de longa distância por fibra óptica e introduziram serviços de alta largura de banda, como ADSL (Asymmetric Digital Subscriber Line). Essas empresas também interromperam sua antiga prática de cobrar preços artificialmente elevados a usuários de serviços de longa distância, a fim de subsidiar o serviço local.

Subitamente, as conexões terrestres de fibra pareciam ser a melhor opção a longo prazo. Apesar disso, os satélites de comunicações têm alguns segmentos de mercado muito importantes que a fibra óptica não é capaz de alcançar. Agora, examinaremos alguns desses mercados.

Em primeiro lugar, apesar de uma única fibra ter, em princípio, maior largura de banda potencial do que todos os satélites lançados até hoje, essa largura de banda não está disponível para a maioria dos usuários. As fibras que estão sendo instaladas atualmente são usadas no sistema telefônico para tratar diversas chamadas interurbanas ao mesmo tempo, não para fornecer uma alta largura de banda a usuários individuais. Com os satélites, torna-se prático um usuário instalar uma antena no telhado do prédio e ignorar por completo o sistema telefônico para obter alta largura de banda. O Teledesic se baseia nessa idéia.

Um segundo nicho de mercado é o da comunicação móvel. Muitas pessoas hoje em dia querem se comunicar enquanto fazem jogging, dirigem, velejam e voam. Os enlaces terrestres de fibra óptica não estão disponíveis para essas pessoas, mas os enlaces de satélite estão potencialmente disponíveis. Porém, é possível que uma combinação de rádio celular e fibra funcione para a maioria dos usuários (mas, provavelmente, não para aqueles que estão sendo transportados

de avião ou estão no mar).

Um terceiro nicho se relaciona a situações em que a difusão é essencial. Uma mensagem enviada por satélite pode ser recebida por milhares de estações terrestres ao mesmo tempo. Por exemplo, uma empresa que transmite um fluxo de preços de ações, apólices ou mercadorias a milhares de corretores deve considerar que um sistema de satélite é mais econômico que similar a difusão no solo.

Um quarto mercado é a comunicação em lugares com terreno inadequado ou com uma infra-estrutura terrestre pouco desenvolvida. Por exemplo, a Indonésia possui seu próprio satélite para o tráfego telefônico doméstico. Lançar um satélite era mais econômico que estender milhares de cabos submarinos entre as 13.677 ilhas do arquipélago.

Um quinto mercado para os satélites tem o objetivo de cobrir áreas em que a obtenção do direito de estender cabos de fibra é difícil ou excessivamente dispendiosa. No sexto nicho, quando a exploração rápida tem importância crítica, como nos sistemas de comunicação militares em tempo de guerra, os satélites ganham com facilidade.

Resumindo, parece que a comunicação do futuro será feita por fibras ópticas terrestres combinadas com rádio celular, mas para algumas aplicações específicas, os satélites são melhores. Entretanto, existe um motivo que se aplica a tudo isso: a economia. Embora a fibra ofereça mais largura de banda, é muito provável que a comunicação terrestre e por satélite entre em uma concorrência agressiva por melhores preços. Se os avanços tecnológicos reduzirem radicalmente o custo de exploração de um satélite (por exemplo, no futuro alguém pode lançar dezenas de satélite de uma só vez), ou se os satélites de baixa órbita se desenvolverem, não é certo que a fibra vencerá em todos os mercados.

[T2] 2.5 A rede pública de telefonia comutada

Quando dois computadores de uma mesma empresa ou organização instalados perto um do outro precisam se comunicar, geralmente é mais fácil conectá-los através de um cabo. As LANs funcionam dessa forma. No entanto, quando as distâncias começam a ficar grandes, há muitos computadores ou os cabos têm de atravessar uma estrada ou outra passagem pública, os custos de instalação de cabos privados costumam ser proibitivos. Além disso, em quase todos os países do mundo, também é ilegal estender linhas de transmissão privadas em (ou sob) propriedades do governo. Conseqüentemente, os projetistas de rede devem utilizar os recursos de telecomunicações existentes.

Esses recursos, em particular a **PSTN (Public Switched Telephone Network)**, foram projetados há muitos anos, tendo em vista um objetivo completamente diferente: a transmissão da voz humana de uma forma mais ou menos reconhecível.

Quando esses recursos são adaptados para a comunicação computador/computador, o resultado é, no máximo, sofrível; porém, com a introdução das fibras ópticas e da tecnologia digital, essa situação está mudando rapidamente. Em qualquer situação, o sistema telefônico está tão estreitamente ligado às redes de computadores (geograficamente distribuídas), que vale a pena dedicarmos uma parte considerável de nosso tempo a estudá-lo.

Para termos uma idéia da magnitude do problema, faremos uma comparação das propriedades de uma conexão computador/computador por um cabo local e por uma linha telefônica de discagem. Um cabo que liga dois computadores pode transferir dados a 10^9 bps, talvez mais. Em contraste, uma linha de discagem tem uma taxa de dados máxima de 56 kbps, uma diferença de quase 20.000 vezes. Essa é a diferença entre o andar gingado de um pato na grama e um foguete para a lua. Se a linha de discagem for substituída por uma conexão ADSL, ainda haverá

uma diferença de 1000 a 2000 vezes.

É óbvio que o problema é o fato de que os projetistas dos sistemas de computadores estão acostumados a trabalhar com esse tipo de tecnologia e, quando se defrontaram com outro sistema cujo desempenho (do ponto de vista deles) é três ou quatro ordens de magnitude pior, eles começam a dedicar muito tempo e esforço tentando descobrir um meio de usá-lo com maior eficiência. Nas próximas seções, descreveremos o sistema telefônico e mostraremos como ele funciona. Para obter informações adicionais sobre o funcionamento interno do sistema telefônico, consulte (Bellamy, 2000).

[T3] 2.5.1 Estrutura do sistema telefônico

Logo depois que Alexander Graham Bell patenteou a invenção do telefone em 1876 (apenas algumas horas antes de seu concorrente, Elisha Gray), havia uma grande demanda por essa nova invenção. Inicialmente, o mercado estava voltado para a venda de telefones, que eram comercializados aos pares. Era o usuário quem tinha de conectar os dois aparelhos usando um fio. Os elétrons eram retornados através do solo. Se quisesse usar o aparelho para conversar com *n* outros proprietários de telefone, o proprietário de um telefone tinha de conectar fios em todas as *n* residências. Em um ano, as cidades ficaram tomadas por fios que passavam pelas casas e pelas árvores, criando um cenário de total desorganização. Logo ficou óbvio que o modelo de conexão de um telefone a outro, como é mostrado na Figura 2.20(a), não funcionaria.

[arte: ver original p. 119]

[Dísticos]

[1] (a) (b) (c)

[F]Figura 2.20

[FL] (a) Rede totalmente interconectada. (b) Switch centralizado. (c) Hierarquia de

dois níveis

Bell percebeu essa situação e criou a Bell Telephone Company, que abriu sua primeira estação de comutação (em New Haven, Connecticut) em 1878. A empresa ligava um fio até a casa ou o escritório de cada usuário. Para fazer uma chamada, o usuário girava a manivela, o que emitia um som na companhia telefônica e chamava a atenção de um operador. Este, por sua vez, conectava manualmente o emissor da chamada ao receptor usando um jumper. Observe na Figura 2.20(b) o modelo de uma única estação de comutação.

Não demorou muito tempo para as estações de comutação da Bell System se espalharem por todos os locais. Logo as pessoas passaram a querer fazer chamadas interurbanas. Por isso, a Bell System passou a conectar uma estação de comutação à outra. Contudo, o problema original veio à tona mais uma vez: conectar cada estação de comutação à outra através de um fio entre elas logo se tornou inviável. Então, as estações de comutação de segundo nível foram inventadas. Depois de algum tempo, tornaram-se necessárias várias estações de segundo nível, como mostra a Figura 2.20(c). Mais tarde, a hierarquia cresceu até alcançar cinco níveis.

Em 1890, era possível notar a presença das três principais partes do sistema telefônico: as estações de comutação, os fios que ligavam os usuários às estações de comutação (agora já operando com cabos de pares trançados, isolados e balanceados em vez de cabos abertos com retorno por terra) e as conexões de longa distância existentes entre as estações de comutação. Apesar de ter havido inúmeros avanços nessas três áreas, o modelo básico da Bell System permaneceu praticamente intacto por mais de 100 anos. Para obter informações técnicas e resumidas sobre o sistema telefônico e sua história, consulte (Hawley, 1991).

Antes da quebra da AT&T em 1984, o sistema telefônico encontrava-se

organizado como uma hierarquia de vários níveis extremamente redundante.
Embora seja bastante simplificada, a descrição apresenta a idéia básica do sistema telefônico. Cada telefone contém dois fios de cobre que saem do aparelho e se conectam diretamente à **estação final** mais próxima da companhia telefônica (também denominada **estação central local**). Em geral, a distância varia de 1 a 10 km, sendo menor nas cidades que nas regiões rurais. Só nos EUA, existem cerca 22.000 estações finais. As conexões através de dois fios entre o telefone de cada assinante e a estação final são conhecidas no mercado como **loop local**. O comprimento de todos os loops locais existentes no mundo inteiro, se eles fossem esticados de uma extremidade à outra, equivaleria a mil vezes a distância da Terra à Lua e de volta à Terra.

Houve uma época em que 80% do capital da AT&T estavam no cobre dos loops locais, o que a tornava a maior mina de cobre do mundo. Felizmente, essa informação não era muito difundida na comunidade financeira. Se tivessem conhecimento desse fato, alguns empresários comprariam a AT&T, terminariam com todos os serviços de telefonia dos EUA, descascariam toda a fiação e venderiam os fios a uma refinaria de cobre para ter um retorno rápido do capital.

Se um assinante conectado a determinada estação final ligar para outro assinante da mesma estação, o mecanismo de comutação dentro da estação irá configurar uma conexão elétrica direta entre os dois loops locais. Essa conexão permanecerá intacta durante a chamada.

Se o telefone chamado estiver conectado a uma outra estação final, outro procedimento terá de ser usado. Cada estação final contém uma série de linhas de saída para um ou mais centros de comutação vizinhos, denominados **estações interurbanas** (ou, se estiverem na mesma área, **estações tandem**). Essas linhas são denominadas **troncos de conexão interurbana.** Se as estações finais do transmissor e do receptor tiverem um tronco de conexão interurbana ligado à

mesma estação interurbana (uma situação bastante provável caso eles estejam geograficamente próximos), a conexão poderá ser estabelecida dentro da estação interurbana. Observe na Figura 2.20(c) uma rede telefônica formada apenas por telefones (os pontos pequenos), estações finais (os pontos maiores) e estações interurbanas (os quadrados).

Se o transmissor e o receptor não compartilharem a mesma estação interurbana, o caminho terá de ser estabelecido em um ponto mais alto da hierarquia. Existem as estações principais, locais e regionais que formam uma rede através da qual as estações interurbanas estão conectadas. As estações interurbanas, principais, locais e regionais, se comunicam entre si através de **troncos interurbanos** de alta largura de banda (também denominados **troncos entre estações**). O número de diferentes tipos de centros de comutação e sua topologia (por exemplo, duas estações locais possuem uma conexão direta ou têm de passar por uma estação regional?) varia de país para país, dependendo da densidade telefônica de cada território. A Figura 2.21 mostra como uma conexão de média distância pode ser roteada.

[arte: ver original p. 121]

[Dísticos]

[1] Telefone Estação final Estação interurbana

Loop local Tronco de conexão interurbana

[2] Estação(ões) intermediária(s) de comutação

Troncos interurbanos de largura de banda muito alta

[3] Estação interurbana Estação final Telefone

Tronco de conexão interurbana Loop local

[F]Figura 2.21

[FL] Rota de um circuito típico para uma chamada de média distância

Nas telecomunicações, são usados vários meios de transmissão. Hoje em dia, os loops locais são formados por cabos de pares trançados da categoria 3. No entanto, nos primórdios da telefonia, o mais comum eram os cabos sem isolamento, separados 25 cm um do outro em postes telefônicos. Entre as estações de comutação, o uso de cabos coaxiais, microondas e principalmente de fibras ópticas era bastante freqüente.

No passado, a transmissão em todo o sistema telefônico era analógica, com o sinal de voz sendo transmitido como uma voltagem elétrica da origem até o destino. Com o advento da fibra óptica, da eletrônica digital e dos computadores, todos os troncos e switches são agora digitais, deixando o loop local como o último fragmento de tecnologia analógica no sistema. A transmissão digital é preferida porque, em uma chamada intercontinental, não é necessário reproduzir com precisão uma forma de onda analógica depois de ter passado por centenas de amplificadores. Ser capaz de distinguir corretamente 0 de 1 já é suficiente. Essa propriedade torna a transmissão digital mais confiável que a analógica. Ela também é mais econômica e de mais fácil manutenção.

Em suma, o sistema telefônico é formado por três componentes principais:

1. Loops locais (pares trançados analógicos indo para as residências e para as empresas).
2. Troncos (fibra óptica digital conectando as estações de comutação).
3. Estações de comutação (onde as chamadas são transferidas de um tronco para outro).

Depois de uma rápida análise da política das companhias telefônicas, voltaremos a cada um desses três componentes e os analisaremos em detalhes. Os loops locais oferecem acesso ao sistema inteiro por todas as pessoas; assim, eles são críticos. Infelizmente, eles também constituem o elo mais fraco no sistema. Para os troncos de longa distância, a principal questão é reunir várias chamadas e

transmiti-las ao mesmo tempo, pela mesma fibra. Esse assunto é chamado multiplexação, e estudaremos três maneiras diferentes de realizar esse processo. Por último, existem duas formas fundamentalmente distintas de executar a comutação; portanto, analisaremos ambas.

[T3] 2.5.2 A política das companhias telefônicas

Por muitas décadas até 1984, a Bell System foi a responsável pelo serviço de chamadas locais e interurbanas nos EUA. Na década de 1970, o governo norte-americano concluiu que esse era um monopólio ilegal e promoveu uma ação para desmembrá-lo. O governo foi vitorioso e, em 1º de janeiro, a AT&T foi dividida na AT&T Long Lines, em 23 **BOCs (Bell Operating Companies)** e em algumas outras partes. As 23 BOCs foram agrupadas em sete BOCs regionais (RBOCs), o que as tornou economicamente viáveis. Toda a natureza do sistema de telecomunicações norte-americano foi alterada da noite para o dia por uma ordem judicial (e *não* por um ato do Congresso dos EUA).

Os detalhes exatos dessa ruptura foram descritos no conhecido **MFJ (Modified Final Judgement**, um paradoxo, se é que houve um — se o julgamento podia ser modificado, isso significava que ele não era o resultado final). Esse fato provocou o aumento da concorrência, a melhoria dos serviços e a redução dos preços para os consumidores e as empresas. Entretanto, os preços para o serviço local cresceram à medida que os subsídios cruzados das chamadas de longa distância foram eliminados, e o serviço local teve de ser tornar auto-suficiente. Hoje em dia, muitos outros países estão considerando a abertura à concorrência em termos semelhantes.

Para determinar precisamente a quem cabiam as responsabilidades, o território dos EUA foi dividido em 164 **LATAs (Local Access and Transport Areas)**. De uma forma bem genérica, uma LATA corresponde à região coberta por um único

código de área. Em geral, dentro de uma LATA existia uma **LEC (Local Exchange Carrier)** que detinha o monopólio do sistema telefônico convencional dentro de sua área. As LECs mais importantes eram as BOCs, embora algumas LATAs contivessem uma ou mais das 1500 companhias telefônicas independentes que operavam como LECs.

Todo o tráfego entre LATAS era manipulado por um tipo diferente de empresa, uma **IXC (IntereXchange Carrier)**. Originalmente, a AT&T Long Lines era a única IXC segura, mas hoje em dia a WorldCom e a Sprint são concorrentes estáveis atuando no mesmo ramo das IXCs. Uma das preocupações ao ocorrer o desmembramento foi assegurar que todas as IXCs seriam tratadas igualmente em termos de qualidade das linhas, das tarifas e do número de dígitos que seus clientes teriam de discar para usá-las. Observe na Figura 2.22 como essa situação é tratada. Nessa figura, vemos três exemplos de LATAs, cada uma com várias estações finais. As LATAs 2 e 3 também têm uma pequena hierarquia com estações tandem (estações interurbanas intraLATA).

Qualquer IXC que deseje se encarregar de chamadas provenientes de uma LATA pode criar uma estação de comutação denominada **POP (Point of Presence — ponto de presença)**. A LEC é necessária para conectar cada IXC a cada estação final, seja diretamente, como nas LATAs 1 e 3, ou indiretamente, como na LATA 2. Além disso, as condições da conexão, tanto técnicas quanto financeiras, têm de ser idênticas para todas as IXCs. Dessa forma, um assinante da LATA 1, por exemplo, pode escolher qual IXC usará para entrar em contato com assinantes que façam parte da LATA 3.

Como parte do MFJ, as IXCs foram proibidas de prestar serviços telefônicos locais, e as LECs foram proibidas de prestar serviços telefônicos interLATAs, apesar de todas serem livres para atuar em quaisquer outros ramos como, por exemplo, pequenos restaurantes. Em 1984, essa era uma condição razoavelmente

não ambígua. Infelizmente, a tecnologia tem uma forma interessante de tornar a lei obsoleta. Nem a TV a cabo nem os telefones celulares foram cobertos pelo acordo. À medida que a TV a cabo passou de unidirecional para bidirecional e a popularidade dos telefones celulares explodiu, as LECs e as IXCs começaram a comprar ou a se associar às operadoras de TV a cabo ou de telefones celulares.

[arte: ver original p. 123]

[Dísticos]

[1] Estação interurbana da IXC 1 Estação interurbana da IXC 2

[2] LATA 1

[3] Para loops locais

LATA 2

[4] POP de IXC

Estação tandem

Estação final

LATA 3

[F]Figura 2.22

[FL] O relacionamento entre LATAs, LECs e IXCs. Todos os círculos são estações de comutação de LECs. Cada hexágono pertence à IXC indicada pelo número

Em 1995, o Congresso dos EUA percebeu que tentar manter uma distinção entre os vários tipos de empresas não era mais sustentável e elaborou um projeto de lei que permitiria às empresas de TV a cabo, companhias telefônicas locais, concessionárias de comunicação de longa distância e operadoras de sistemas celulares a entrarem nos ramos de negócios umas das outras. A idéia era que qualquer empresa poderia oferecer a seus clientes um único pacote integrado contendo serviços de TV a cabo, de telefone e de informações, e que diferentes empresas seriam concorrentes em serviços e preços. O projeto de lei foi

sancionado em fevereiro de 1996. Como resultado, algumas BOCs se tornaram IXCs e algumas outras empresas, como operadoras de TV a cabo, começaram a oferecer serviços de telefonia local, competindo com as LECs.

Uma propriedade interessante da lei de 1996 é a exigência de que as LECs implementem portabilidade de número local. Isso significa que um cliente pode mudar de companhias telefônica local sem ter de receber um novo número de telefone. Essa providência remove um enorme obstáculo para muitas pessoas e as torna muito mais inclinadas a mudar de LECs, aumentando assim a concorrência. Como resultado, o cenário das telecomunicações dos Estados Unidos está passando atualmente por uma reestruturação radical. Mais uma vez, muitos outros países estão começando a seguir o mesmo caminho. Com freqüência, outros países esperam para ver como esse tipo de experiência funciona nos EUA. Se der certo, eles fazem o mesmo; se funcionar mal, eles algo diferente.

[T3] 2.5.3 O loop local: modems, ADSL e redes sem fios

Agora chegou o momento de iniciarmos nosso estudo detalhado do funcionamento do sistema de telefonia. As principais partes do sistema estão ilustradas na Figura 2.23. Aqui vemos os loops locais, os troncos, e também as estações interurbanas e as estações finais, que contêm equipamentos para realizar a comutação das chamadas. Uma estação final tem até 10.000 loops locais (nos Estados Unidos e em outros países grandes). De fato, até recentemente, a união do código de área com o código da operadora indicava a estação final; então, (212) 601-xxxx representava uma estação final específica com 10.000 assinantes, numerados de 0000 até 9999. Com o advento da concorrência pelo serviço local, esse sistema deixou de ser viável, porque várias empresas queriam ter a propriedade do código da estação final. Além disso, o número de códigos foi basicamente esgotado, e assim foi necessário introduzir

complexos esquemas de mapeamento.

Vamos começar pela parte com que a maioria das pessoas está familiarizada: o loop local de dois fios que vem da estação final de uma companhia telefônica até residências e pequenas empresas. Com freqüência, o loop local também é chamado "o último quilômetro", embora o comprimento real possa chegar a vários quilômetros. Ele utiliza sinalização analógica há mais de 100 anos e é provável que continue a utilizá-la por mais alguns anos, devido ao custo elevado da conversão para sinalização digital. Não obstante, mesmo nesse último bastião da transmissão analógica, estão ocorrendo mudanças. Nesta seção, estudaremos o loop local tradicional e os novos desenvolvimentos que estão surgindo, com ênfase particular na comunicação de dados a partir de computadores domésticos.

[arte: ver original p. 124]

[Dísticos]

[1] Computador

Loop local (analógico, par trançado)

Modem

[2] Tronco de largura de banda média (digital, fibra)

Codec

Estação final

[3] Estação interurbana

Estação interurbana Estação interurbana

Tronco de alta largura de banda (digital, fibra)

[4] ISP 2

Linha digital

Até 10.000 loops locais

[5] Banco de modems

Codec

ISP 1

[F]Figura 2.23

[FL] O uso das transmissões analógica e digital para uma chamada entre dois computadores. A conversão é feita por modems e codecs

Quando um computador deseja transmitir dados digitais por uma linha de discagem analógica, primeiro os dados devem ser convertidos para a forma analógica, a fim de se realizar a transmissão pelo loop local. Essa conversão é feita por um dispositivo chamado **modem**, algo que estudaremos em breve. Na estação final da companhia telefônica, os dados são convertidos para a forma digital, a fim de serem transmitidos pelos troncos de longa distância.

Se na outra extremidade houver um computador com um modem, a conversão inversa — de digital para analógico — será necessária para o sinal percorrer o loop local no destino. Essa organização é mostrada na Figura 2.23 para o ISP (Internet Service Provider — provedor de serviços da Internet) 1, que tem um banco de modems, cada um deles conectado a um loop local diferente. Esse ISP pode lidar com tantas conexões quantos forem os modems (supondo-se que seu servidor ou servidores tenha(m) capacidade de computação suficiente). Essa organização era a normal até surgirem os modems de 56 kbps, por razões que se tornarão claras em breve.

A sinalização analógica consiste na variação de uma voltagem com o tempo para representar um fluxo de informações. Se os meios de transmissão fossem perfeitos, o receptor receberia exatamente o mesmo sinal que o transmissor enviou. Infelizmente, os meios não são perfeitos. Por isso, o sinal recebido não é igual ao sinal transmitido. No caso de dados digitais, essa diferença pode provocar erros.

As linhas de transmissão enfrentam três problemas principais: atenuação,

distorção de retardo e ruído. A **atenuação** é a perda de energia, à medida que o sinal se propaga externamente. A perda é expressa em decibéis por quilômetro. A quantidade de energia perdida varia em função da freqüência. Para ver o efeito dessa variação em função da freqüência, imagine um sinal não como uma forma de onda simples, mas como componentes de uma série de Fourier. Cada componente é atenuado em uma proporção diferente, o que resulta em um espectro de Fourier diferente no receptor.

Para piorar ainda mais, os diversos componentes de Fourier também se propagam em velocidades diferentes no fio. Essa diferença de velocidade leva à **distorção** do sinal recebido na outra extremidade.

Outro problema é o **ruído**, que consiste em energia indesejável proveniente de outras fontes que não o transmissor. O ruído térmico é causado pelo movimento aleatório dos elétrons em um fio, e é inevitável. A linha cruzada é provocada pelo acoplamento indutivo entre dois fios que estão próximos um do outro. Às vezes, quando fala ao telefone, você pode ouvir outra conversação no fundo, o que chamamos de linha cruzada. Por fim, existe o ruído de impulso, que é provocado, dentre outros fatores, por picos de voltagem na linha de energia. No caso de dados digitais, o ruído de impulso pode provocar a perda de um ou mais bits.

[T4] Modems

Devido aos problemas que acabamos de analisar, e principalmente ao fato de a atenuação e a velocidade de propagação variarem em função da freqüência, não é interessante ter uma grande variedade de freqüências no sinal. Infelizmente, as ondas quadradas utilizadas em sinais digitais têm um amplo espectro de freqüência e, portanto, estão sujeitas a uma forte atenuação e a uma distorção de retardo. Esses efeitos tornam a sinalização de banda base (DC) inadequada, exceto em baixas velocidades e em distâncias curtas.

Para contornar os problemas associados à sinalização DC, principalmente nas linhas telefônicas, é usada a sinalização AC. É introduzido no sinal um tom contínuo na faixa de 1.000 a 2.000 Hz, denominado **portadora de onda senoidal**. Sua amplitude, freqüência ou fase pode ser modulada para transmitir informações. Na **modulação de amplitude**, duas amplitudes diferentes são usadas para representar 0 e 1, respectivamente. Na **modulação de freqüência**, também conhecida como **frequency shift keying** (chaveamento por deslocamento de freqüência), são usados dois (ou mais) tons diferentes. (O termo **keying** — ou **chaveamento** — também é amplamente utilizado na indústria como sinônimo de modulação.) Na forma mais simples da **modulação de fase**, a onda portadora é deslocada de forma sistemática 0 ou 180 graus em intervalos uniformemente espaçados. Um esquema melhor é usar deslocamentos de 45, 135, 225 ou 315 graus para transmitir dois bits de informações por intervalo de tempo. Além disso, a exigência de sempre ocorrer um deslocamento de fase no fim de cada intervalo de tempo facilita o reconhecimento dos limites dos intervalos de tempo pelo receptor.

[arte: ver original p. 126]

[Dísticos]

[1] 0 1 0 1 1 0 0 1 0 0 1 0 0

[2] Mudanças de fase

[F]Figura 2.24

[FL] (a) Um sinal binário. (b) Modulação de amplitude. (c) Modulação de freqüência. (d) Modulação de fase

A Figura 2.24 ilustra as três formas de modulação. Na Figura 2.24(a) uma das amplitudes é diferente de zero e uma é igual a zero. Na Figura 2.24(b), são usadas duas freqüências. Na Figura 2.24(c), um deslocamento de fase está

presente ou ausente no limite de cada bit. Um dispositivo que aceita um fluxo serial de bits como entrada e produz uma portadora modulada por um (ou mais) desses métodos (ou vice-versa) é chamado **modem** (modulador-demodulador). O modem é inserido entre o computador (digital) e o sistema telefônico (analógico). Para atingir velocidades cada vez mais altas, não basta apenas aumentar a taxa de amostragem. O teorema de Nyquist afirma que mesmo com uma linha de 3.000 Hz perfeita (e a linha telefônica definitivamente não se enquadra nessa categoria), não há razão para uma amostragem mais rápida que 6.000 Hz. Na prática, a maioria dos modems realiza amostragens 2400 vezes/segundo e se concentra em obter mais bits por amostra.

O número de amostras por segundo é medido em **baud**. Durante cada baud, é enviado um **símbolo**. Desse modo, uma linha de *n* bauds transmite *n* símbolos/s. Por exemplo, uma linha de 2400 bauds envia um símbolo a cada 416.667 µs. Se o símbolo consiste em 0 volts para indicar um valor 0 lógico e 1 volt para representar um valor 1 lógico, a taxa de bits é 2400 bps. Porém, se as voltagens 0, 1, 2, e 3 volts são usadas, cada símbolo consiste em 2 bits, e assim uma linha de 2400 bauds pode transmitir 2400 símbolos/s a uma taxa de dados de 4800 bps. De modo semelhante, com quatro deslocamentos de fase possíveis, também existem 2 bits/símbolo, e portanto mais uma vez a taxa de bits é o dobro da taxa em bauds. Essa última técnica é amplamente usada e se denomina **QPSK (Quadrature Phase Shift Keying — chaveamento por deslocamento de fase de quadratura)**.

Os conceitos de largura de banda, baud, símbolo e taxa de bits costumam ser confundidos; assim, vamos redefini-los aqui. A largura de banda de um meio é a faixa de freqüências que passam por ele com atenuação mínima. É uma propriedade física do meio (normalmente, variando de 0 até alguma freqüência máxima) e é medida em Hz. A taxa de bauds é o número de amostras/s

realizadas. Cada amostra envia um fragmento de informações, ou seja, um símbolo. A taxa de bauds e a taxa de símbolos são portanto idênticas. A técnica de modulação (por exemplo, QPSK) determina o número de bits/símbolo. A taxa de bits é a quantidade de informações enviadas pelo canal e é igual ao número de símbolos/s multiplicado pelo número de bits/símbolo.

Todos os modems avançados utilizam uma combinação de técnicas de modulação para transmitir vários bits por baud. Com freqüência, várias amplitudes e vários deslocamentos de fase são combinados para transmitir diversos bits/símbolo. Na Figura 2.25(a), podemos observar pontos a 45, 135, 225 e 315 graus com amplitude constante (distância a partir da origem). A fase de um ponto é indicada pelo ângulo que uma linha dele até a origem forma com o eixo x positivo. A Figura 2.25(a) tem quatro combinações válidas e pode ser usada para transmitir dois bits por símbolo. Ela é QPSK.

Na Figura 2.25(b), vemos uma outra estrutura de modulação, na qual são usadas quatro amplitudes e quatro fases, dando um total de 16 combinações diferentes. Esse esquema de modulação pode ser usado para transmitir quatro bits por símbolo. Ele é chamado **QAM-16 ((Quadrature Amplitude Modulation — modulação por amplitude de quadratura)**. Às vezes, o termo **16-QAM** também é usado. A QAM-16 pode ser empregada para transmitir, por exemplo, 9600 bps por uma linha de 2400 bauds.

A Figura 2.25(c) é outro esquema de modulação envolvendo amplitude e fase. Ele permite 64 combinações diferentes, de forma que podem ser transmitidos seis bits por símbolo. Ele é chamado **QAM-64**. Também são usadas QAMs de ordem mais alta.

[arte: ver original p. 128]

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem

seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 2.25

[FL] (a) QPSK. (b) QAM-16. (c) QAM-64

Diagramas como os da Figura 2.25, que mostram combinações legítimas de amplitude e fase, são denominados **diagramas de constelação.** Cada padrão de modem de alta velocidade contém seu próprio padrão de constelação e pode se comunicar apenas com outros modems que utilizem o mesmo padrão (embora a maioria dos modems possa emular todos os padrões mais lentos).

Com muitos pontos no padrão de constelação, até mesmo uma pequena quantidade de ruído na amplitude ou fase detectada pode resultar em um erro e, potencialmente, em muitos bits incorretos. Para reduzir a chance de um erro, os padrões para os modems de velocidade mais alta efetuam a correção de erros adicionando bits extras a cada amostra. Os esquemas são conhecidos como **TCM (Trellis Coded Modulation — modulação codificada por treliças).** Desse modo, por exemplo, o padrão de modem V.32 utiliza 32 pontos de constelação para transmitir 4 bits de dados e 1 bit de paridade por símbolo a 2400 bauds, a fim de alcançar 9600 bps com correção de erros. Seu padrão de constelação é mostrado na Figura 2.26(a). A decisão de "girar" em torno da origem 45 graus foi tomada por razões de engenharia; as constelações giradas e não giradas têm a mesma capacidade de informações.

O próximo passo acima de 9600 bps é 14.400 bps. Ele é chamado **V.32 bis.** Essa velocidade é alcançada transmitindo-se 6 bits de dados e 1 bit de paridade por amostra a 2400 bauds. Seu padrão de constelação tem 128 pontos quando é usada a QAM-128, e está ilustrado na Figura 2.26(b). Os fax modems utilizam essa velocidade para transmitir páginas digitalizadas como mapas de bits. A QAM-256 não é usada em qualquer modem de telefone padrão, mas é empregada

em redes a cabo, como veremos.

O modem de telefone que segue o V.32 bis é o **V.34**, que funciona em 28.800 bps a 2400 bauds com 12 bits de dados/símbolo. O último modem dessa série é o **V.34 bis** que utiliza 14 bits de dados/símbolo a 2400 bauds para atingir 33.600 bps.

Para aumentar ainda mais a taxa de dados efetiva, muitos modems compactam os dados antes de transmiti-los, a fim de obter uma taxa de dados efetiva maior que 33.600 bps. Por outro lado, quase todos os modems testam a linha antes de começar a transmitir dados do usuário e, se descobrirem que a qualidade é deficiente, reduzem a velocidade para um valor mais baixo que o valor máximo nominal. Desse modo, a velocidade *efetiva* do modem observada pelo usuário pode ser mais baixa, igual ou mais alta que a velocidade nominal oficial.

[arte: ver original p. 129]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 2.26

[FL] (a) V.32 para 9600 bps. (b) V.32 bis para 14400 bps

Todos os modems modernos permitem tráfego em ambos os sentidos ao mesmo tempo (usando freqüências diferentes para sentidos diferentes). Uma conexão que permite tráfego em ambos os sentidos simultaneamente é chamada **full-duplex**. Uma estrada de duas pistas é full-duplex. Uma conexão que permite o tráfego nos dois sentidos, mas apenas em um sentido de cada vez, é chamada **half-duplex**. Uma estrada de ferro única é half-duplex. Uma conexão que permite o tráfego apenas em um sentido é chamada **simplex**. Uma rua de mão única é simplex. Outro exemplo de uma conexão simplex é uma fibra óptica com um

laser em uma extremidade e um detector de luz na outra extremidade.

A razão pela qual os modems padrão param em 33.600 é que o limite de Shannon para o sistema telefônico é de aproximadamente 35 kbps, e assim uma transmissão mais rápida que isso violaria os leis da física (departamento de termodinâmica). Para descobrir se os modems de 56 kbps são teoricamente possíveis, fique atento.

Porém, por que o limite teórico é de 35 kbps? Ele está relacionado com o comprimento médio dos loops locais e com a qualidade dessas linhas. O limite de 35 kbps é determinado pelo comprimento médio dos loops locais. Na Figura 2.23, uma chamada originada no computador da esquerda e encerrada no ISP 1 passa por dois loops locais como um sinal analógico, uma vez na origem e uma vez no destino. Cada um desses loops locais acrescenta ruído ao sinal. Se pudéssemos nos livrar de um desses loops locais, a taxa máxima seria duplicada. O ISP 2 faz exatamente isso. Ele recebe um fluxo digital puro da estação final mais próxima. O sinal digital usado nos troncos é entregue diretamente aos ISP 2, eliminando os codecs, os modems e a transmissão analógica em sua extremidade. Desse modo, quando uma extremidade da conexão é puramente digital, como ocorre com a maioria dos ISPs atuais, a taxa máxima de dados pode chegar a 70 kbps. Entre dois usuários domésticos com modems e linhas analógicas, o máximo é 33,6 kbps.

A razão para o uso de modems de 56 kbps está relacionada ao teorema de Nyquist. O canal telefônico tem cerca de 4000 Hz de largura (incluindo as bandas de proteção). O número máximo de amostras independentes por segundo é portanto 8000. O número de bits por amostra nos Estados Unidos é 8, um dos quais é usado para fins de controle, permitindo 56.000 bits/s de dados do usuário. Na Europa, todos os 8 bits estão disponíveis para os usuários, e assim poderiam ser usados modems de 64.000 bits/s; porém, para se chegar a um

acordo internacional sobre um padrão, foi escolhido o valor 56.000.

Esse padrão de modem é chamado **V.90**. Ele proporciona um canal upstream (do usuário para o ISP) de 33,6 kbps, mas um canal downstream (do ISP para o usuário) de 56 kbps, porque em geral existe mais transporte de dados do ISP para o usuário do que no sentido inverso (por exemplo, a solicitação de uma página da Web exige apenas alguns bytes, mas a página real pode ter megabytes). Na teoria, seria possível um canal upstream com largura maior que 33,6 kbps mas, como muitos loops locais são ruidosos demais até mesmo para 33,6 kbps, decidiu-se alocar uma parte maior da largura de banda para o canal downstream, a fim de aumentar as chances de ele funcionar realmente a 56 kbps.

O próximo passo além do V.90 é o **V.92**. Esses modems são capazes de transmitir 48 kbps no canal upstream, se a linha puder lidar com isso. Eles também determinam a velocidade apropriada a usar em cerca de metade dos 30 segundos habituais exigidos pelos modems mais antigos. Finalmente, eles permitem que uma chamada telefônica recebida interrompa uma sessão da Internet, desde que a linha tenha um serviço de espera por chamadas.

[T4] Linhas digitais do assinante

Quando a indústria de telefonia finalmente conseguiu alcançar a 56 kbps, ela se congratulou pelo serviço bem feito. Enquanto isso, a indústria de TV a cabo estava oferecendo velocidades de até 10 Mbps sobre cabos compartilhados, e as empresas de satélites estavam planejando oferecer mais de 50 Mbps. À medida que o acesso à Internet se tornou uma parte cada vez mais importante de seus negócios, as companhias telefônicas (LECs) começaram a perceber que precisavam de um produto mais competitivo. Sua resposta foi começar a oferecer novos serviços digitais sobre o loop local. Os serviços com maior largura de banda que o serviço de telefonia padrão costumam ser chamados serviços de

banda larga, embora a expressão seja mais um conceito de marketing que um conceito técnico específico.

Inicialmente, havia muitas ofertas sobrepostas, todas sob o nome genérico **xDSL (Digital Subscriber Line — linha digital do assinante)**, para diversos *x*. Descreveremos a seguir essas ofertas, mas vamos nos concentrar principalmente naquele que provavelmente se tornará o mais popular desses serviços, o **ADSL (Asymmetric DSL)**. Tendo em vista que o ADSL ainda está sendo desenvolvido e nem todos os padrões estão plenamente estabelecidos, alguns dos detalhes que apresentaremos podem mudar com o tempo, mas o quadro básico deve permanecer válido. Para obter mais informações sobre o ADSL, consulte (Summers, 1999; e Vetter *et al.*, 2000).

A razão para os modems serem tão lentos é que os telefones foram inventados para transportar a voz humana, e o sistema inteiro foi cuidadosamente otimizado para esse propósito. Os dados sempre estiveram em segundo plano. No ponto em que cada loop local termina na estação final, o fio passa por um filtro que atenua todas as freqüências abaixo de 300 Hz e acima de 3400 Hz. O corte não é nítido — 300 Hz e 3400 Hz são os pontos de 3 dB — assim, a largura de banda é mencionada normalmente como 4000 Hz, embora a distância entre os pontos de 3 dB seja de 3100 Hz. Portanto, os dados também estão restritos a essa banda estreita.

O artifício que faz o xDSL funcionar é o fato de que, quando um cliente se inscreve nele, a linha de entrada é conectada a um tipo diferente de switch, que não tem esse filtro, tornando assim disponível toda a capacidade do loop local. Então, o fator limitador passa a ser a constituição física do loop local, não a largura de banda artificial de 3100 Hz criada pelo filtro.

Infelizmente, a capacidade do loop local depende de vários fatores, incluindo seu comprimento, sua espessura e sua qualidade geral. A Figura 2.27 mostra um

esboço da largura de banda potencial como uma função da distância. Essa figura pressupõe que todos os outros fatores estão otimizados (novos fios, pacotes modestos etc.).

[arte: ver original p. 131]

[Dísticos]

[1]Mbps

50

40

30

20

10

0

[2]0 1000 2000 3000 4000 5000 6000

Metros

[F]Figura 2.27

[FL] Variação da largura de banda com a distância sobre o UTP da categoria 3 para DSL

A implicação dessa figura cria um problema para a companhia telefônica. Quando escolhe uma velocidade para oferecer, ela está ao mesmo tempo escolhendo um raio a partir de suas estações finais, além do qual o serviço não poderá ser oferecido. Isso significa que, quando clientes distantes tentarem assinar o serviço, eles receberão a seguinte mensagem: "Muito obrigado por seu interesse, mas você está 100 metros além da distância máxima da central mais próxima que poderia lhe oferecer o serviço. Você não gostaria de mudar?" Quanta mais baixa a velocidade escolhida, maior o raio e maior o número de clientes cobertos. Porém, quanto mais baixa a velocidade, menos atraente será o serviço e menor o número

de pessoas que estarão dispostas a pagar por ele. É aqui que os negócios encontram a tecnologia. (Uma solução potencial é construir miniestações finais nas vizinhanças, mas essa é uma proposta dispendiosa.)

Todos os serviços xDSL foram criados visando a certos objetivos. Primeiro, os serviços devem funcionar nos loops locais de pares trançados da categoria 3 existente. Segundo, eles não devem afetar os telefones e os aparelhos de fax atuais dos clientes. Em terceiro lugar, eles devem ser muito mais rápidos que 56 kbps. Em quarto lugar, eles devem estar sempre ativos, apenas com uma tarifa mensal, mas nenhuma tarifa por minuto.

A oferta inicial do ADSL foi feita pela AT&T e funcionava dividindo o espectro disponível no loop local, cerca de 1,1 MHz, em três bandas de freqüência: **POTS (Plain Old Telephone Service)**, upstream (do usuário para a estação final) e downstream (da estação final para o usuário). A técnica de ter várias bandas de freqüência é chamada multiplexação por divisão de freqüência; vamos estudá-la em detalhes em uma seção posterior. As ofertas subseqüentes de outros provedores adotaram uma abordagem diferente e parece que essa última deverá ser vitoriosa; assim, vamos descrevê-la a seguir.

A abordagem alternativa, chamada **DMT (Discrete MultiTone)**, está ilustrada na Figura 2.28. Na verdade, ela divide o espectro de 1,1 MHz disponível no loop local em 256 canais independentes de 4312,5 Hz cada. O canal 0 é usado para o POTS. Os canais de 1 a 5 não são usados, a fim de impedir que o sinal de voz e os sinais de dados interfiram uns com os outros. Dos 250 canais restantes, um é utilizado para o controle upstream e outro é empregado para o controle downstream. Os outros canais estão disponíveis para dados do usuário.

[arte: ver original p. 132]

[Dísticos]

[1] 256 canais de 4 kHz

[2] Energia

[3] 0 25 1100 kHz

Voz Upstream Downstream

[F]Figura 2.28

[FL] Operação do ADSL usando modelação discreta multitonal (de vários tons)

Em princípio, cada um dos canais restantes pode ser usado em um fluxo de dados full-duplex; porém, harmônicos, linhas cruzadas e outros efeitos impedem a utilização de sistemas práticos bem abaixo do limite teórico. Cabe ao provedor definir quantos canais serão usados para upstream e para downstream. Uma mistura de 50% para cada um é tecnicamente possível, mas a maioria dos provedores aloca algo como 80% a 90% da largura de banda ao canal downstream, pois a maioria dos usuários faz mais download do que upload de dados. Essa escolha deu origem à letra "A" no acrônimo ADSL. Uma divisão comum reserva 32 canais para upstream e os restantes para downstream. Também é possível tornar bidirecionais alguns dos canais upstream mais altos para aumentar a largura de banda, embora essa otimização exija o uso de um circuito especial para cancelamento de eco.

O padrão ADSL (ANSI T1.413 e ITU G.992.1) permite velocidades de até 8 Mbps downstream e 1 Mbps upstream. Porém, poucos provedores oferecem essa velocidade. Em geral, os provedores oferecem 512 kbps downstream e 64 kbps upstream (serviço padrão) e 1 Mbps downstream e 256 kbps upstream (serviço especial).

Dentro de cada canal, é usado um esquema de modulação semelhante ao V.34, embora a taxa de amostragem seja de 4000 bauds, em vez de 2400 bauds. A qualidade da linha em cada canal é monitorada constantemente, e a taxa de dados é ajustada de forma contínua quando necessário, de modo que diferentes

canais possam ter taxas de dados distintas. Os dados reais são enviados com modulação QAM, com até 15 bits por baud, usando um diagrama de constelação análogo ao da Figura 2.25(b). Por exemplo, com 224 canais downstream e 15 bits/baud a 4000 bauds, a largura de banda downstream é 13,44 Mbps. Na prática, a relação sinal/ruído nunca é boa o bastante para se alcançar essa taxa, mas é possível utilizar 8 Mbps por curtos períodos sobre loops de alta qualidade; foi por essa razão que o padrão chegou tão longe.

A Figura 2.29 mostra uma organização ADSL típica. Nesse esquema, um técnico da companhia telefônica deve instalar um **NID (Network Interface Device — dispositivo de interface de rede**) no local do cliente. Essa pequena caixa plástica marca o fim da propriedade da companhia telefônica e o início da propriedade do cliente. Próximo ao NID (ou às vezes combinado a ele) há um **divisor**, um filtro analógico que separa a banda de 0 a 4000 Hz utilizada pelo POTS dos dados. O sinal do POTS é roteado até o telefone ou o equipamento de fax existente, e o sinal de dados é roteado até um modem ADSL. Na realidade, o modem ADSL é um processador de sinais digitais configurado para atuar como 250 modems QAM operando em paralelo em freqüências diferentes. Tendo em vista que a maioria dos modems ADSL atuais é externa, o computador deve estar conectado ao modem em alta velocidade. Normalmente, isso é feito inserindo-se uma placa Ethernet no computador e operando-se uma Ethernet muito de dois nós muito curta, contendo apenas o computador e o modem ADSL. Ocasionalmente, é usada a porta USB em lugar da conexão Ethernet. Sem dúvida, estarão disponíveis no futuro placas de modem ADSL internas.

[arte: ver original p. 133]

[Dísticos]

[1] Switch de voz

Codec

Divisor

DSLAM

Para o ISP

Estação final da companhia telefônica

[2] Linha telefônica

[3] Telefone

Divisor

NID

Computador

Modem ADSL

Ethernet

Local do cliente

[F]Figura 2.29

[FL]Uma configuração típica de equipamento ADSL

Na outra extremidade do fio, no lado da estação final, está instalado um divisor correspondente. Aqui, a porção de voz do sinal é filtrada e enviada ao switch de voz normal. O sinal acima de 26 kHz é roteado para um novo tipo de dispositivo, chamado **DSLAM (Digital Subscriber Line Access Multiplexer — multiplexador de acesso à linha digital do assinante)**, que contém a mesma espécie de processador de sinais digitais que o modem ADSL. Uma vez que o sinal digital é recuperado em um fluxo de bits, são formados pacotes que são enviados ao ISP.

Essa separação completa entre o sistema de voz e o ADSL torna relativamente fácil para uma companhia telefônica distribuir o serviço ADSL. Basta adquirir um DSLAM e um divisor, e conectar os assinantes do ADSL ao divisor. Outros serviços de alta largura de banda (por exemplo, ISDN) exigem mudanças muito maiores no equipamento de comutação existente.

Uma desvantagem do projeto da Figura 2.29 é a presença do NID e do divisor no local do cliente. A instalação desses itens só pode ser feita por um técnico da companhia telefônica, necessitando de um dispendioso serviço de assistência (isto é, enviar um técnico até o local do cliente). Portanto, também teve de ser padronizado um projeto alternativo sem divisores, Informalmente, ele é chamado G.lite, mas o número do padrão ITU é G.992.2. Ele é idêntico ao da figura 2.29, mas não em o divisor. A linha telefônica existente é usada como está. A única diferença é a inserção de um microfiltro em cada tomada de telefone, entre o telefone ou o modem ADSL e o fio. O microfiltro para o telefone é um @@@filtro de banda baixa, que elimina freqüências acima de 3400 Hz; o microfiltro para o modem ADSL é um @@@filtro de banda alta, que elimina freqüências abaixo de 26 kHz. Porém, esse sistema não é tão confiável quanto um divisor, e assim G.lite só pode ser usado até 1,5 Mbps (contra 8 Mbps para o ADSL com um divisor). Contudo, G.lite ainda exige um divisor na estação final, mas essa instalação não requer um grande número de serviços de assistência.

O ADSL é apenas um padrão da camada física. O que funciona sobre ele depende da portadora. Freqüentemente, a opção é o ATM, devido à capacidade do ATM para administrar a qualidade do serviço e ao fato de que muitas companhias telefônicas executam o ATM na rede de núcleo.

[T4] Loops locais sem fios

Desde 1996 nos Estados Unidos e um pouco mais tarde em outros países, as empresas que desejam competir com a companhia telefônica local fortificada (detentora do monopólio das comunicações), denominada **ILEC (Incumbent LEC)**, são livres para fazê-lo. As candidatas mais prováveis são empresas de telefonia de longa distância (IXCs). Qualquer IXC que deseje entrar no negócio de telefonia local em alguma cidade deve realizar certas ações. Primeiro, ela tem de comprar

ou alugar um edifício como sua primeira estação final nessa cidade. Em segundo lugar, ela deve ocupar a estação final com switches telefônicos e outros equipamentos, todos disponíveis como produtos prontos de diversos fornecedores. Em terceiro lugar, ela deve estender um cabo de fibra entre a estação final e sua estação interurbana mais próxima, de forma que os novos clientes locais tenham acesso à sua rede nacional. Em quanto lugar, ela deve buscar clientes, em geral anunciando um serviço melhor ou preços mais baixos que os da ILEC.

Em seguida, começa a parte difícil. Vamos supor que surjam realmente alguns clientes. Como a nova companhia telefônica local, chamada **CLEC (Competitive LEC)**, irá conectar os telefones e os computadores dos clientes à sua novíssima estação final? Comprar os direitos necessários e estender fios ou fibras é algo proibitivo. Muitas CLECs descobriram uma alternativa mais econômica para o loop local de par trançado tradicional: o **WLL (Wireless Local Loop — loop local sem fio)**.

Em certo sentido, um telefone fixo que usa um loop local sem fio é um pouco parecido com um telefone celular, mas há três diferenças técnicas cruciais. Primeiro, o cliente do loop local sem fio com freqüência deseja conectividade de alta velocidade para a Internet, muitas vezes a velocidades no mínimo iguais às do ADSL. Em segundo lugar, o novo cliente talvez não se importe de ter um técnico da CLEC instalando uma grande antena direcional em seu telhado, apontada para a estação final da CLEC. Em terceiro lugar, o usuário não se movimenta, o que elimina todos os problemas de mobilidade e handoff de células que veremos mais adiante neste capítulo. Assim, nasceu uma nova indústria: a das **redes sem fios fixas** (serviço local de telefonia e da Internet prestado por CLECs através de loops locais sem fios).

Embora as WLLs tenham começado a operar seriamente em 1998, temos de voltar

até 1969 para conhecer sua origem. Nesse ano, a FCC alocou dois canais de televisão (a 6 MHz cada) como televisão educativa a 2,1 GHz. Em anos subseqüentes, foram acrescentados mais 31 canais a 2,5 GHz, perfazendo um total de 198 MHz.

A televisão educativa nunca decolou e, em 1998, a FCC aceitou a devolução das freqüências e as alocou ao rádio bidirecional. De imediato, elas foram ocupadas por loops locais sem fios. A essas freqüências, as microondas têm 10 a 12 cm de comprimento. Elas têm um alcance de cerca de 50 km e podem penetrar moderadamente na vegetação e na chuva. Os 198 MHz do novo espectro foram imediatamente postos em uso nos loops locais sem fios, sob a forma de um serviço chamado **MMDS (Multichannel Multipoint Distribution Service — serviço de distribuição multiponto multicanal)**. O MMDS pode ser considerado uma MAN (Metropolitan Area Network — rede metropolitana), da mesma forma que seu primo LMDS (que discutiremos em seguida).

A grande vantagem desse serviço é que a terminologia está bem estabelecida e o equipamento prontamente disponível. A desvantagem é que a largura de banda total disponível é modesta, e tem de ser compartilhada por muitos usuários espalhados por uma região geográfica bastante grande.

A baixa largura de banda do MMDS fez aumentar o interesse em ondas milimétricas, como uma alternativa. Em freqüências entre 28 e 31 GHz nos Estados Unidos e 40 GHz na Europa, nenhuma freqüência foi alocada, porque era difícil construir circuitos integrados de silício que operem tão rápido. Esse problema foi resolvido com a criação de circuitos integrados de arsenieto de gálio, abrindo as bandas milimétricas à comunicação por rádio. A FCC respondeu à demanda, alocando 1,3 GHz a um novo serviço de loop local sem fios chamado **LMDS (Local Multipoint Distribution Service — serviço de distribuição multiponto local)**. Essa alocação representa o maior bloco isolado de largura de banda já

alocada pela FCC para qualquer uso. Um bloco semelhante está sendo alocado na Europa, mas a 40 GHz.

A operação do LMDS é mostrada na Figura 2.30. Nessa figura, está ilustrada uma torre com várias antenas, cada uma apontada para uma direção diferente. Tendo em vista que as ondas milimétricas são altamente direcionais, cada antena define um setor, independente dos outros. Nessa freqüência, o intervalo é de 2 a 5 km, o que significa que são necessárias muitas torres para cobrir a área de uma cidade.

[arte: ver original p. 136]

[Dísticos]

[1] Rede telefônica

[2] ISP

[F]Figura 2.30

[FL] Arquitetura de um sistema LMDS

Como o ADSL, o LMDS utiliza uma alocação de largura de banda assimétrica que favorece o canal downstream. Com tecnologia atual, cada setor pode ter 36 Gbps downstream e 1 Mbps upstream, compartilhados entre todos os usuários desse setor. Se cada usuário ativo baixar três páginas de 5 KB por minuto, o usuário estará ocupando uma média de 2000 bps do espectro, o que permite um máximo de 18.000 usuários ativos por setor. Porém, para manter o retardo razoável, o serviço deve manter no máximo 9.000 usuários ativos. Com quatro setores, como mostra a Figura 2.30, poderia ser admitida uma população de 36.000 usuários ativos. Supondo que um em cada três clientes esteja on-line durante os períodos de pico, uma única torre com quatro antenas poderia atender a 100.000 pessoas dentro de um raio de 5 km da torre. Esses cálculos foram feitos por muitas CLECs potenciais, algumas das quais concluíram que, com um investimento modesto em

torres de ondas milimétricas, elas poderiam entrar no negócio de telefonia local e de Internet, oferecendo aos usuários taxas de dados comparáveis às de TV a cabo, a um preço mais baixo.

Porém, o LMDS tem alguns problemas. Por um lado, as ondas milimétricas se propagam em linha reta, e assim deve haver uma linha de visão desimpedida entre as antenas instaladas no telhado e a torre. Por outro lado, as folhas absorvem bem essas ondas, e portanto a torre deve ser alta o bastante para evitar a presença de árvores na linha de visão. Além disso, o que pode parecer uma linha de visão desimpedida em julho talvez não esteja tão desimpedida em dezembro, quando as árvores estão cheias de folhas. A chuva também absorve essas ondas. Até certo ponto, os erros introduzidos pela chuva podem ser compensados com o uso de códigos de correção de erros ou aumentando-se a potência quando estiver chovendo. Apesar disso, é mais provável que o serviço LMDS se desenvolva primeiro em climas secos, como no nordeste, e não na Amazônia, uma região mais úmida.

É pouco provável que os loops locais sem fios se tornem populares, a menos que existam padrões para incentivar os fornecedores de equipamentos a fabricarem produtos e para assegurar que os clientes poderão trocar de CLECs sem terem de adquirir novos equipamentos. Para fornecer essa padronização, o IEEE instalou um comitê chamado 802.16 para definir um padrão de LMDS. O padrão 802.16 foi publicado em abril de 2002. O IEEE chama o padrão 802.16 de **MAN sem fio.**

O IEEE 802.16 foi projetado para telefonia digital, acesso à Internet, conexão de duas LANs remotas, difusão de televisão e rádio, e outros usos. Examinaremos esse padrão com mais detalhes no Capítulo 4.

[T3] 2.5.4 Troncos e multiplexação

As economias em escala desempenham um importante papel no sistema

telefônico. Em essência, o custo para instalar e manter um tronco de alta largura de banda é o mesmo de um tronco de baixa largura de banda entre duas estações de comutação (ou seja, os custos são decorrentes da instalação em si e não do uso de fios de cobre ou de fibra óptica). Como conseqüência, as companhias telefônicas desenvolveram esquemas elaborados para multiplexar muitas conversações em um único tronco físico. Esses esquemas de multiplexação podem ser divididos em duas categorias básicas: **FDM (Frequency Division Multiplexing — multiplexação por divisão de freqüência)** e **TDM (Time Division Multiplexing — multiplexação por divisão de tempo).** Na FDM, o espectro de freqüência é dividido em bandas de freqüência, tendo cada usuário a posse exclusiva de alguma banda. Na TDM, os usuários se revezam (em um esquema de rodízio), e cada um obtém periodicamente a largura de banda inteira por um determinado período de tempo.

A transmissão de rádio AM serve de ilustração para ambos os tipos de multiplexação. O espectro alocado é de cerca de 1 MHz, aproximadamente 500 a 1.500 kHz. Diferentes freqüências são alocadas a diferentes canais lógicos (estações), cada um operando em uma parte do espectro, sendo a separação entre canais grande o bastante para evitar interferência. Esse sistema é um exemplo de multiplexação por divisão de freqüência. Além disso (em alguns países), as estações individuais têm dois subcanais lógicos: música e propaganda. Eles se alternam na mesma freqüência, primeiro um período de música, depois um período de publicidade, depois mais música e assim por diante. Essa situação representa a multiplexação por divisão de tempo.

Vamos examinar a seguir a multiplexação por divisão de freqüência. Depois veremos como a FDM pode ser aplicada a fibras ópticas (multiplexação por divisão de comprimento de onda). Em seguida, passaremos à TDM e terminaremos com um sistema TDM avançado utilizado em fibras ópticas

(SONET).

[T4] Multiplexação por divisão de freqüência

A Figura 2.31 mostra como três canais telefônicos de nível de voz são multiplexados com o uso da FDM. Os filtros limitam a largura de banda utilizável a cerca de 3100 Hz por canal de qualidade de voz. Quando muitos canais são multiplexados ao mesmo tempo, são alocados 4000 Hz para cada canal, a fim de mantê-los bem separados. Primeiro, os canais de voz têm sua freqüência aumentada, cada qual com um valor diferente. Depois eles podem ser combinados, pois agora não há dois canais ocupando a mesma porção do espectro. Observe que, apesar de haver intervalos (bandas de proteção) entre os canais, há uma certa sobreposição entre canais adjacentes, porque os filtros não têm limites nítidos. Essa sobreposição significa que um forte pico no limite de um canal será sentido no canal adjacente como ruído não térmico.

[arte: ver original p. 138]

[Dísticos]

[1]Fator de atenuação

[2]Canal 1

1

Canal 2

1

Canal 3

1

300 3100

Freqüência (Hz)

(a)

[3] 60 64 68 72

Freqüência (kHz)

(b)

[4] Canal 2

Canal 1 Canal 3

60 64 68 72

Freqüência (kHz)

(c)

[F]Figura 2.31

[FL] Multiplexação por divisão de freqüência. (a) As larguras de banda originais. (b) As larguras de banda aumentaram em freqüência. (c) O canal multiplexado

Os esquemas FDM utilizados em todo o mundo têm um certo grau de padronização. Um padrão muito difundido tem doze canais de voz de 4000 Hz multiplexados na banda de 60 a 108 kHz. Essa unidade é chamada **grupo**. Às vezes, a banda de 12 a 60 kHz é utilizada por outro grupo. Muitas concessionárias de comunicações oferecem aos clientes um serviço de linha privada de 48 a 56 kbps baseado no grupo. Cinco grupos (60 canais de voz) podem ser multiplexados para formar um **supergrupo**. A unidade seguinte é o **grupo mestre**, que tem cinco supergrupos (padrão CCITT) ou dez supergrupos (Bell System). Também existem outros padrões de até 230 mil canais de voz.

[T4] Multiplexação por divisão de comprimento de onda

No caso de canais de fibra óptica, é usada uma variação de multiplexação por divisão de freqüência. Trata-se da **WDM (Wavelength Division Multiplexing — multiplexação por divisão de comprimento de onda)**. O princípio básico da WDM em fibras está representado na Figura 2.32. Aqui, quatro fibras chegam juntas a um combinados óptico, cada uma com sua energia presente em um comprimento

de onda distinto. Os quatro feixes são combinados em uma única fibra compartilhada para transmissão a um destino remoto. Na extremidade remota, o feixe é dividido no mesmo número de fibras que havia no lado da entrada. Cada fibra de saída contém um núcleo curto especialmente construído que filtra todos os comprimentos de onda com exceção de um. Os sinais resultantes podem ser roteados até seu destino ou recombinados de diferentes maneiras para transporte multiplexado adicional.

[arte: ver original p. 139]

[Dísticos]

[1]Espectro da fibra 1

Energia

λ

[2]Espectro da fibra 2

Energia

λ

[3]Espectro da fibra 3

Energia

λ

[4]Espectro da fibra 4

Energia

λ

[5]Espectro na fibra compartilhada

Energia

λ

[6]

Fibra 1 λ_1

Fibra 2 λ_2

Fibra 3 λ_3 Combinador

Fibra 4 λ_4

[7] $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$

Fibra compartilhada de longa distância

[8]Filtro

λ_2

λ_4

Divisor λ_1

λ_3

[F]Figura 2.32

[FL] Multiplexação por divisão de comprimento de onda

Realmente não há nada de novo aqui. Trata-se apenas da multiplexação por divisão de freqüência em freqüências muito altas. Desde que cada canal tenha sua própria faixa de freqüências (isto é, de comprimentos de onda) e todas as faixas sejam disjuntas, elas poderão ser multiplexadas na fibra de longa distância. A única diferença em relação à FDM elétrica é que um sistema óptico que utilize uma grade de difração será completamente passivo e, portanto, altamente confiável.

A tecnologia WDM tem progredido a uma velocidade que deixa envergonhada a tecnologia de informática. A WDM foi criada por volta de 1990. Os primeiros sistemas comerciais tinham oito canais, com 2,5 Gbps por canal. Em 1998, os sistemas com 40 canais de 2,5 Gbps estavam no mercado. Em 2001, havia produtos com 96 canais de 10 Gbps, dando um total de 960 Gbps. Essa largura de banda é suficiente para transmitir 30 filmes de longa metragem por segundo (em MPEG-2). Sistemas com 200 canais já estão funcionando em laboratório. Quando o número de canais é muito grande e os comprimentos de onda estão

pouco espaçados — por exemplo, 0,1 nm — o sistema costuma ser chamado **DWDM (Dense WDM).**

Devemos observar que a razão para o WDM ser popular é o fato de a energia em uma única fibra ter normalmente apenas alguns gigahertz de largura, porque no momento é impossível realizar a conversão entre meios físicos elétricos e ópticos com rapidez maior que essa. Utilizando-se muitos canais em paralelo com diferentes comprimentos de onda, a largura de banda agregada aumenta de forma linear com o número de canais. Como a largura de banda de uma única banda de fibra é aproximadamente 25.000 GHz (veja a Figura 2.6), teoricamente existe espaço para 2500 canais de 10 Gbps, mesmo a 1 bit/Hz (e também são possíveis taxas mais altas).

Outra novidade é o desenvolvimento de amplificadores totalmente ópticos. Antes, a cada 100 km era necessário dividir todos os canais e converter cada um deles em um sinal elétrico para amplificação separada, antes de convertê-los novamente em sinais ópticos e combiná-los. Hoje, os amplificadores totalmente ópticos podem regenerar o sinal inteiro uma única vez a cada 1000 km, sem a necessidade de várias conversões ópticas/elétricas.

No exemplo da Figura 2.32, temos um sistema de comprimento de onda fixo. Bits da fibra de entrada 1 vão para a fibra de saída 3, bits da fibra de entrada 2 vão para a fibra de saída 1 etc. Porém, também é possível criar sistemas WDM comutados. Em dispositivos como esses, os filtros de saída são ajustáveis com o uso de interferômetros de Fabry-Perot ou Mach-Zehnder. Para obter mais informações sobre a WDM e sua aplicação à comutação de pacotes da Internet, consulte (Elmirghani e Mouftah, 2000; Hunter e Andonovic, 2000; e Listani *et al.*, 2001).

[T4] Multiplexação por divisão de tempo

A tecnologia WDM é maravilhosa, mas ainda existe muito fio de cobre no sistema telefônico; assim, vamos voltar a ele por enquanto. Embora a FDM ainda seja usada com fios de cobre ou canais de microondas, ela exige circuitos analógicos e não é adequada para uso por um computador. Em contraste, a TDM pode ser inteiramente manipulada por circuitos eletrônicos digitais; portanto, ela se tornou muito mais difundida nos últimos anos. Infelizmente, ela só pode ser usada para dados digitais. Como os loops locais produzem sinais analógicos, uma conversão de analógico para digital se faz necessária na estação final, onde todos os loops locais individuais chegam juntos para serem combinados em troncos de saída.

Agora, vamos examinar como vários sinais de voz analógicos são digitalizados e combinados em um único tronco digital de saída. Os dados de computadores enviados por um modem também são analógicos; assim, a descrição apresentada a seguir também se aplica a eles. Os sinais analógicos são digitalizados na estação final por um dispositivo chamado **codec** (codificador-decodificador), produzindo uma série de números de 8 bits. O codec cria 8000 amostras por segundo (125 μs/amostra), pois o teorema de Nyquist diz que isso é suficiente para captar todas as informações da largura de banda do canal telefônico de 4 kHz. Em uma taxa de amostragem mais baixa, as informações se perderiam; a uma taxa mais alta, nenhuma informação extra seria obtida. Essa técnica é chamada **PCM (Pulse Code Modulation — modulação por código de pulso)**. A PCM forma o núcleo do sistema telefônico moderno. Como conseqüência, virtualmente todos os intervalos de tempo no sistema telefônico são múltiplos de 125 μs.

Quando a transmissão digital começou a surgir como tecnologia viável, o CCITT foi incapaz de chegar a um acordo sobre um padrão internacional para PCM. Conseqüentemente, agora existe uma variedade de esquemas incompatíveis em uso em diferentes países no mundo.

O método em uso na América do Norte e no Japão é a portadora **T1**, representada

na Figura 2.33. (Tecnicamente falando, o formato é chamado DS1, e a portadora é chamada T1 mas, seguindo a tradição da indústria, não faremos aqui essa sutil distinção.) A portadora T1 consiste em 24 canais de voz multiplexados juntos. Em geral, é feita uma amostragem dos sinais analógicos em rodízio, e o fluxo analógico resultante é enviado para o codec, em vez de serem utilizados 24 codecs separados para depois mesclar a saída digital. Por sua vez, cada um dos 24 canais consegue inserir 8 bits no fluxo de saída. Sete bits representam dados, e um é usado para controle, produzindo 7 × 8000 = 56.000 bps de dados e 1 × 8000 = 8000 bps de informações de sinalização por canal.

[arte: ver original p. 141]

[Dísticos]

[1]Quadro de 193 bits (125 µs)

[2]Canal 1 Canal 2 Canal 3 Canal 4 Canal 24

[3]O bit 1 é um código de enquadramento

[4]7 bits de dados por canal em cada amostra

[5]O bit 8 é usado para sinalização

[F]Figura 2.33

[FL] A portadora T1 (1,544 Mbps)

Um quadro consiste em 24 × 8 = 192 bits, mais um bit extra para enquadramento, produzindo 193 bits a cada 125 µs. Isso resulta em uma taxa de dados bruta de 1,544 Mbps. O 193º bit é usado para sincronização de quadros e utiliza o padrão 0101010101. Normalmente, o receptor continua a conferir esse bit para garantir que não perdeu a sincronização. Se sair de sincronismo, o receptor poderá procurar por esse padrão para se ressincronizar. Clientes analógicos não podem gerar o padrão de bits, pois ele corresponde a uma onda senoidal a 4000 Hz, que seria filtrada. Clientes digitais podem, é claro, gerar esse

padrão, mas não é provável que estejam presentes quando houver algum problema com o quadro. Quando um sistema T1 está sendo utilizado inteiramente para dados, apenas 23 dos canais são utilizados para esse fim. O 24º canal é empregado para um padrão especial de sincronização, a fim de permitir a recuperação mais rápida no caso de problemas com o quadro.

Quando o CCITT finalmente chegou a um acordo, percebeu que 8000 bps de informações de sinalização era muito; portanto, seu padrão de 1,544 Mbps se baseou em um item de dados de 8 bits, e não de 7bits; ou seja, o sinal analógico é quantizado em 256, e não em 128 níveis discretos. Duas variações (incompatíveis) são fornecidas. Na **sinalização por canal comum**, o bit extra (anexado ao final e não ao início do quadro de 193 bits) utiliza os valores 10101010 ... nos quadros ímpares e contém informações de sinalização para todos os canais nos quadros pares.

Na outra variação, a **sinalização por canal associado**, cada canal tem seu próprio subcanal de sinalização. Um subcanal privado é organizado alocando-se um dos oito bits do usuário a cada seis quadros para fins de sinalização; portanto, cinco das seis amostras têm 8 bits de largura, e a outra tem apenas 7 bits de largura. O CCITT também recomendou uma portadora PCM a 2,048 Mbps, chamada **E1**. Essa portadora tem 32 amostras de dados de 8 bits compactadas no quadro básico de 125 µs. Trinta dos canais são utilizados para informações, e dois são empregados na sinalização. Cada grupo de quatro quadros fornece 64 bits de sinalização, metade dos quais é usada para a sinalização por canal associado e a outra metade é usada para sincronização de quadros, ou é reservada por cada país para utilização livre. Fora da América do Norte e do Japão, a portadora E1 de 2,048 Mbps é usada em lugar da portadora T1.

Uma vez digitalizado, o sinal de voz tenta usar técnicas estatísticas para reduzir o número de bits necessários por canal. Essas técnicas são apropriadas não apenas

para codificação de voz, mas também para a digitalização de qualquer sinal analógico. Todos os métodos de compactação se baseiam no princípio de que o sinal muda de forma relativamente lenta em comparação com a freqüência de amostragem; portanto, grande parte das informações do nível digital de 7 ou 8 bits é redundante.

Um método, chamado **modulação de código de pulso diferencial (differential pulse code modulation)** ou PCM diferencial, consiste em gerar como saída não a amplitude digitalizada, mas sim a diferença entre o valor atual e o anterior. Tendo em vista que saltos de ±16 (ou mais) em uma escala de 128 são improváveis, 5 bits deverão ser suficientes em vez de 7. Se ocasionalmente o sinal saltar de forma incontrolável, talvez a lógica de codificação exija vários períodos de amostragem para recuperar o tempo perdido. No caso da voz, o erro introduzido pode ser ignorado.

[arte: ver original p. 142]

[Dísticos]

[1]Amostras consecutivas sempre diferem por ± 1

[2]Níveis de digitalização

15

10

5

0

[3]Sinal alterado rápido demais para a codificação acompanhar

[4] 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1

Tempo

[5]Intervalo de amostragem

[6]Fluxo de bits enviados

[F]Figura 2.34

[FL] Modulação delta

Uma variação desse método de compactação exige que cada valor de amostragem seja diferente de seu predecessor por +1 ou –1. Sob essas condições, um único bit pode ser transmitido, informando se a nova amostra está acima ou abaixo da anterior. Essa técnica, chamada **modulação delta**, está ilustrada na Figura 2.34. A exemplo de todas as técnicas de compactação que pressupõem pequenas alterações de nível entre amostras consecutivas, a codificação delta pode ter problemas se o sinal mudar rápido demais, como mostra a figura. Quando isso ocorre, as informações são perdidas.

Um aperfeiçoamento da PCM diferencial consiste em extrapolar os valores anteriores para prever o valor seguinte, e depois codificar a diferença entre o sinal real e o sinal previsto. É claro que o transmissor e o receptor devem utilizar o mesmo algoritmo de previsão. Esses esquemas são chamados **codificação por previsão**. Eles são úteis porque reduzem o tamanho dos números a serem codificados e, consequentemente, o número de bits a serem enviados.

A multiplexação por divisão de tempo permite que várias portadoras T1 sejam multiplexadas em portadoras de ordem mais alta. A Figura 2.35 mostra como isso pode ser feito. À esquerda, vemos quatro canais T1 sendo multiplexados em um canal T2. A multiplexação em T2 e acima dele é feita bit a bit, em vez de ser realizada byte a byte com os 24 canais de voz que constituem um quadro T1. Quatro fluxos T1 a uma velocidade de 1,544 Mbps deveriam gerar 6,176 Mbps, mas T2 na verdade tem 6,312 Mbps. Os bits extras são usados para enquadramento e recuperação, no caso de a portadora apresentar alguma falha. T1 e T3 são extensamente utilizados pelos clientes, enquanto T2 e T4 são usados apenas dentro do sistema de telefonia propriamente dito, e portanto não são bem conhecidos.

[arte: ver original p. 143]

[Dísticos]

[1] 4 fluxos T1 de entrada 7 fluxos T2 de entrada 6 fluxos T3 de entrada

[2] 1 fluxo T2 de saída

[3]

4 0

5 1

6 2 4:1 6 5 4 3 2 1 0 7:1 6:1

7 3

[4] 1,544 Mbps 6,312 Mbps 44,736 Mbps 274,176 Mbps

[5] T1 T2 T3 T4

[F]Figura 2.35

[FL] Multiplexação de fluxos T1 em portadoras de velocidade mais alta

No nível seguinte, sete fluxos T2 são combinados bit a bit para formar um fluxo T3. Depois, seis fluxos T3 são unidos para formar um fluxo T4. Em cada etapa, um pequeno volume de overhead é adicionado para fins de enquadramento e recuperação, no caso de ser perdida a sincronização entre transmissor e receptor.

Da mesma forma que existe pouco consenso quanto à portadora básica entre os Estados Unidos e o restante do mundo, há igualmente pouco consenso sobre como ela será multiplexada em portadoras de largura de banda mais alta. O esquema americano de avançar por 4, 7 e 6 não foi adotado por mais ninguém; assim, o padrão CCITT requer multiplexação de quatro fluxos em um fluxo a cada nível. Além disso, o enquadramento e a recuperação de dados são diferentes entre os padrões dos EUA e do CCITT. A hierarquia CCITT para 32, 128, 512, 2048 e 8192 canais funciona em velocidades de 2,048, 8,848, 34,304, 139,264 e 565,148 Mbps.

[T4] SONET/SDH

Nos primórdios da fibra óptica, cada companhia telefônica tinha seu próprio sistema óptico TDM patenteado. Depois que a AT&T foi desmembrada em 1984, as companhias telefônicas locais tiveram de se conectar a diversas concessionárias de comunicações de longa distância, todas com diferentes sistemas ópticos TDM, o que tornou óbvia a necessidade de padronização. Em 1985, a Bellcore, a unidade de pesquisa do RBOC, começou a trabalhar em um padrão, denominado **SONET (Synchronous Optical NETwork — rede óptica síncrona)**. Mais tarde, o CCITT também começou a participar desse trabalho, o que resultou em um padrão SONET e em um conjunto de recomendações paralelas do CCITT (G.707, G.708 e G.709) em 1989. As recomendações do CCITT são chamadas **SDH (Synchronous Digital Hierarchy — hierarquia digital síncrona)**, mas só diferem da SONET em pequenos detalhes. Praticamente todo o tráfego telefônico de longa distância nos Estados Unidos e grande parte dele em outros lugares utiliza agora troncos que executam a SONET na camada física. Para obter informações adicionais sobre a SONET, consulte (Bellamy, 2000; Goralski, 2000; e Shepard, 2001).

O projeto SONET tem quatro objetivos principais. Acima tudo, a SONET tinha de tornar possível a interligação de diferentes concessionárias em rede. A concretização desse objetivo exigia a definição de um padrão de sinalização comum relacionado a comprimento de onda, sincronização, estrutura de enquadramento e outras questões.

Em segundo lugar, foram necessários alguns meios para unificar os sistemas digitais dos Estados Unidos, Europa e Japão, todos baseados em canais PCM de 64 kbps, mas todos combinados de formas diferentes (e incompatíveis).

Em terceiro lugar, a SONET teve de proporcionar um modo de multiplexar vários

canais digitais. No momento em que a SONET surgiu, a portadora digital de velocidade mais alta usada em todo território dos Estados Unidos era a T3, a 44,736 Mbps. A T4 já havia sido definida, mas não era muito usada, e nada que ultrapassasse a velocidade da T4 havia sido definido. Parte da missão da SONET era dar continuidade à hierarquia até gigabits/s e proporcionar velocidades ainda maiores. Também era necessária uma forma padrão de multiplexar canais mais lentos em um canal SONET.

Em quarto lugar, a SONET tinha de oferecer recursos de operação, administração e manutenção (OAM). Os sistemas anteriores não faziam isso muito bem.

Uma decisão inicial foi tornar a SONET um sistema TDM tradicional, com toda a largura de banda da fibra dedicada a um único canal contendo slots de tempo para os diversos subcanais. Portanto, a SONET é um sistema síncrono, controlado por um relógio mestre, cuja precisão é de aproximadamente uma parte em 10^9. Os bits em uma linha SONET são transmitidos a intervalos extremamente precisos, controlados pelo relógio mestre. Quando a comutação de células foi proposta mais tarde para servir de base ao ATM, o fato de permitir chegadas de células irregulares fez com que ele fosse identificado como *Asynchronous* Transfer Mode, em contraste com a operação síncrona da SONET. com a SONET, o transmissor e o receptor estão vinculados a um relógio comum; com o ATM, isso não acontece.

O quadro básico da SONET é um bloco de 810 bytes, transmitido a cada 125 µs. Tendo em vista que a SONET é síncrona, os quadros são emitidos independente de haver ou não dados úteis a enviar. A taxa de 8000 quadros/s corresponde exatamente à taxa de amostragem dos canais PCM utilizados em todos os sistemas de telefonia digital.

Os quadros de 810 bytes da SONET são melhor descritos como um retângulo de bytes, com 90 colunas de largura por 9 linhas de altura. Desse modo, $8 \times 810 =$

6480 bits são transmitidos 8 mil vezes por segundo, o que resulta em uma taxa de dados bruta de 51,84 Mbps. Esse é o canal básico da SONET, chamado **STS-1 (Synchronous Transport Signal-1)**. Todos os troncos SONET são múltiplos do STS-1.

As três primeiras colunas de cada quadro são reservadas para as informações de gerenciamento do sistema, conforme ilustra a Figura 2.36. As três primeiras linhas contêm o overhead de seção; as seis linhas seguintes contêm o overhead de linha. O overhead de seção é gerado e verificado no início e no fim de cada seção, enquanto o overhead de linha é gerado e verificado no início e no fim de cada linha.

[arte: ver original p. 145]

[Dísticos]

[1]3 colunas para overhead

9 linhas

[2]87 colunas

[3]Quadro SONET (125 µs)

Quadro SONET (125 µs)

[4]Overhead de seção

[5]Overhead de linha

[6]Overhead de caminho

[7]SPE

[F]Figura 2.36

[FL] Dois quadros duplos na rede SONET

Um transmissor SONET transmite quadros duplos de 810 bytes em seqüência, sem intervalos entre eles, mesmo quando não existem dados (e, nesse caso, ele transmite dados fictícios). Do ponto de vista do receptor, tudo que ele vê é um

fluxo de bits contínuo; assim, como saber onde começa cada quadro? A resposta é que os dois primeiros bytes de cada quadro contêm um padrão fixo que o receptor procura. Se encontra esse padrão no mesmo lugar em um número grande de quadros consecutivos, o receptor pressupõe que está sincronizado com o transmissor. Na teoria, um usuário poderia inserir esse padrão na carga útil de um modo regular; porém, na prática, isso não pode ser feito devido à multiplexação de diversos usuários no mesmo quadro, além de outras razões.

As 87 colunas restantes contêm $87 \times 9 \times 8 \times 8000 = 50,112$ Mbps de dados do usuário. Entretanto, os dados do usuário, chamados **SPE (Synchronous Carga útil Envelope — envelope síncrono de carga útil)** nem sempre começam na linha 1, coluna 4. O SPE pode começar em qualquer lugar do quadro. A primeira linha do overhead de linha contém um ponteiro que indica o primeiro byte. A primeira coluna do SPE é o overhead de caminho (ou seja, o cabeçalho do protocolo da subcamada de caminho fim a fim).

A hierarquia de multiplexação da SONET é mostrada na Figura 2.37. Foram definidas taxas de STS-1 a STS-48. A portadora óptica que corresponde a STS-*n* é chamada OC-*n*; sua configuração bit a bit é a mesma, exceto por uma certa reordenação de bits, necessária para sincronização. Os nomes SDH são diferentes; eles começam em OC-3 porque os sistemas baseados no CCITT não têm uma taxa próxima a 51,84 Mbps. A portadora OC-9 está presente, porque corresponde aproximadamente à velocidade de um tronco de grande porte e alta velocidade usado no Japão. A OC-18 e o OC-36 são utilizados no Japão. A taxa de dados bruta inclui todo o overhead. A taxa de dado SPE exclui o overhead de linha e o de seção. A taxa de dados do usuário exclui todo o overhead e só considera as 86 colunas de carga útil.

[arte: ver original p. 146]

[T]Tabela

SONET		SDH	Taxa de dados (Mbps)		
Elétrica	**Óptica**	**Óptica**	**Bruta**	**SPE**	**Usuário**
STS-1	OC-1		51,84	50,112	49,536
STS-3	OC-3	STM-1	155,52	150,336	148,608
STS-9	OC-9	STM-3	466,56	451,008	445,824
STS-12	OC-12	STM-4	622,08	601,344	594,432
STS-18	OC-18	STM-6	933,12	902,016	891,648
STS-24	OC-24	STM-8	1244,16	1202,688	1188,864
STS-36	OC-36	STM-12	1866,24	1804,032	1783,296
STS-48	OC-48	STM-16	2488,32	2405,376	2377,728
STS-192	OC-192	STM-64	9953,28	9621,504	9510,912

[F]Figura 2.37

[FL] Taxas de multiplexação da SONET e da SDH

Por outro lado, quando uma portadora como a OC-3 não é multiplexada, mas transporta os dados de uma única origem, a letra *c* (significando concatenado) é acrescentada à designação; sendo assim, OC-3 indica uma portadora de 155,52 Mbps composta por três portadoras OC-1 distintas, mas OC-3c indica um fluxo de dados de uma única origem a uma velocidade de 155,52 Mbps. Os três fluxos OC-1 contidos em um fluxo OC-3c são entrelaçados por coluna; primeiro, a coluna 1 do fluxo 1, depois a coluna 1 do fluxo 2, a coluna 1 do fluxo 3, seguida pela coluna 2 do fluxo 1 e assim por diante, resultando em um quadro com 270 colunas de largura e 9 linhas de profundidade.

[T3] 2.5.5 Comutação

Do ponto de vista do engenheiro de telefonia médio, o sistema telefônico é dividido em duas partes principais: a planta externa (os loops locais e troncos,

pois eles estão localizados fisicamente fora das estações de comutação) e a planta interna (os switches), que estão situados no interior das estações de comutação. Acabamos de estudar a planta externa. Chegou a hora de examinarmos a planta interna.

Dentro do sistema telefônico são usadas hoje duas técnicas de comutação diferentes: a comutação de circuitos e a comutação de pacotes. Daremos a seguir uma breve introdução a cada uma delas. Depois, veremos em detalhes a comutação de circuitos, porque é assim que o sistema telefônico atual funciona. Em capítulos subseqüentes, estudaremos minuciosamente a comutação de pacotes.

[T4] Comutação de circuitos

Quando você ou seu computador efetua uma chamada telefônica, o equipamento de comutação do sistema telefônico procura um caminho físico desde o seu telefone até o telefone do receptor. Essa técnica, chamada **comutação de circuitos**, é apresentada esquematicamente na Figura 2.38(a). Cada um dos seis retângulos representa uma estação de comutação da concessionária de comunicações (estação final, estação interurbana etc.). Nesse exemplo, cada estação tem três linhas de entrada e três linhas de saída. Quando uma chamada passa por uma estação de comutação, é (conceitualmente) estabelecida uma conexão física entre a linha que transportou a chamada e uma das linhas de saída, como mostram as linhas pontilhadas.

[arte: ver original p. 147]

[Dísticos]

[1] Conexão física (de cobre) estabelecida quando a chamada é feita

[2] (a) Estação de comutação

[3] Computador

[4] Pacotes enfileirados para transmissão subseqüente

[5] (b) Computador

[F]Figura 2.38

[FL] (a) Comutação de circuitos. (b) Comutação de pacotes

Nos primórdios da telefonia, a conexão era feita pela telefonista que conectava um cabo de ligação em ponte (jumper) aos soquetes de entrada e saída. Na realidade, existe uma pequena história surpreendente associada à invenção do equipamento automático de comutação de circuitos. Esse dispositivo foi inventado por um agente funerário do Século XIX, Almon B. Strowger. Logo depois que o telefone foi inventado, quando uma pessoa morria, alguém ligava para a telefonista da cidade e dizia: "Por favor, ligue-me com um agente funerário". Infelizmente para o Sr. Strowger, havia dois agentes funerários em sua cidade, e a esposa do outro agente era a telefonista da cidade. Ele percebeu rapidamente que teria de inventar um equipamento automático de comutação telefônica ou seu negócio iria à falência. Ele escolheu a primeira opção. Por cerca de 100 anos, o equipamento de comutação de circuitos usado em todo o mundo foi conhecido como engrenagem de Strowger. (A história não registra se a telefonista, agora desempregada, conseguiu emprego como operadora de informações, respondendo a perguntas como: "Qual é o número do telefone do agente funerário?")

O modelo mostrado na Figura 2.39(a) é altamente simplificado, porque partes do caminho físico entre os dois telefones podem de fato ser enlaces de microondas ou de fibra, nos quais são multiplexados milhares de chamadas. Entretanto, a idéia básica é válida: uma vez estabelecida uma chamada, haverá um caminho dedicado entre ambas as extremidades, e ele continuará a existir até que a chamada seja finalizada.

A alternativa para a comutação de circuitos é a comutação de pacotes, mostrada na Figura 2.38(b). Com essa tecnologia, pacotes individuais são enviados conforme necessário, sem a configuração com antecedência de qualquer caminho dedicado. Cabe a cada pacote descobrir sozinho seu caminho até o destino.

Uma propriedade importante da comutação de circuitos é a necessidade de se estabelecer um caminho fim a fim, *antes* que qualquer dado possa ser enviado. O tempo decorrido entre o fim da discagem e o momento em que o telefone começa a tocar pode chegar a 10 segundos ou mais em chamadas interurbanas ou internacionais. Durante esse intervalo de tempo, o sistema telefônico procura uma conexão física, como mostra a Figura 2.39(a). Observe que, antes mesmo de se iniciar a transmissão de dados, o sinal de solicitação de chamada deve se propagar em todo o trajeto até o destino e lá ser reconhecido. Para muitas aplicações de informática (por exemplo, a verificação do crédito em um ponto de venda), tempos de configuração longos são indesejáveis.

Como conseqüência do caminho reservado entre o transmissor e o receptor da chamada, uma vez estabelecida a configuração, o único atraso para a entrega dos dados é o tempo de propagação do sinal eletromagnético, cerca de 5 ms por 1000 km. Outra conseqüência do caminho estabelecido é que não há perigo de congestionamento — ou seja, quando a chamada é feita, você nunca obtém sinais de ocupado. É claro que seria possível receber um sinal de ocupado antes do estabelecimento da conexão, devido à falta de capacidade de comutação ou de troncos.

[T4] Comutação de mensagens

Uma estratégia alternativa de comutação é a **comutação de mensagens**, mostrada na Figura 2.39(b). Quando essa forma de comutação é usada, nenhum caminho físico é estabelecido com antecedência entre o transmissor e o receptor. Em vez

disso, quando o transmissor tem um bloco de dados a ser enviado, esse bloco é armazenado na primeira estação de comutação (isto é, no roteador) e depois é encaminhado, um hop de cada vez. Cada bloco é recebido integralmente, inspecionado em busca de erros, e depois retransmitido. Uma rede que utiliza essa técnica é chamada **store-and-forward**, conforme mencionamos no Capítulo 1.

[arte: ver original p. 149]

[Dísticos]

[1] Sinal de solicitação de chamada

[2] Tempo

Tempo gasto na busca de um tronco de saída

Sinal de aceitação de chamada

[3] Dados

Tronco AB Tronco BC Tronco CD

A B C D

(a)

[4] Retardo de propagação Mensagem

Mensagem Retardo de enfileiramento

Mensagem

[5] A B C D

(b)

[6] Pacote 1

Pacote 2 Pacote 1

Pacote 3 Pacote 2 Pacote 1

Pacote 3 Pacote 2

Pacote 3

[7] A B C D

(c)

[F]Figura 2.39

[FL] Sincronização de eventos em (a) comutaçdo de circuitos, (b) comutação de mensagens, (c) comutação de pacotes

Os primeiros sistemas eletromecânicos de telecomunicações utilizavam a comutação de mensagens, especificamente para telegramas. A mensagem era perfurada em uma fita de papel (off-line) na estação transmissora e, em seguida, era lida e transmitida através de uma linha de comunicações para a próxima estação do trajeto, onde era perfurada em uma fita de papel. A fita era rasgada por um operador e lida em uma das muitas leitoras de fita, uma para cada tronco de saída. Uma estação de comutação como essa era chamada **estação de fita cortada.** A fita de papel não existe mais, e a comutação de mensagens deixou de ser usada; assim, não a discutiremos mais neste livro.

[T4] Comutação de pacotes

Com a comutação de mensagens, não há nenhum limite sobre o tamanho do bloco, o que significa que os roteadores (em um sistema moderno) devem ter discos para armazenar temporariamente no buffer blocos longos. Isso também significa que um único bloco pode obstruir uma linha entre roteadores por alguns minutos, tornando a comutação de mensagens inútil para o tráfego interativo. Para contornar esses problemas, foi inventada a **comutação de pacotes**, descrita no Capítulo 1. As redes de comutação de pacotes impõem um limite máximo restrito sobre o tamanho do bloco, permitindo que os pacotes sejam ar-mazenados temporariamente na memória principal do roteador e não em um disco. Assegurando que nenhum usuário poderá monopolizar uma linha de transmissão durante muito tempo (milissegundos), as redes de comutação de

pacotes se adequam bem à manipulação de tráfego interativo. Outra vantagem da comutação de pacotes sobre a comutação de mensagens é mostrada na Figura 2.39(b) e (c): o primeiro pacote de uma mensagem com vários pacotes pode ser encaminhado antes do segundo ter chegado completamente, o que reduz o retardo e melhora a velocidade de transferência (o throughput). Por isso, em geral as redes de computadores utilizam a técnica de comutação de pacotes. Ocasionalmente, elas utilizam a comutação de circuitos, mas nunca a comutação de mensagens.

A comutação de circuitos e a comutação de pacotes diferem em muitos aspectos. Para começar, a comutação de circuitos exige que um circuito seja configurado de ponta a ponta antes de se iniciar a comunicação. A comutação de pacotes não exige qualquer configuração antecipada. O primeiro pacote pode ser enviado assim que está disponível.

O resultado da instalação de conexão com comutação de circuitos é a reserva de largura de banda em todo o percurso, desde o transmissor até o receptor. Todos os pacotes seguem esse caminho. Entre outras propriedades, fazer todos os pacotes seguirem o mesmo caminho significa que eles não poderão chegar fora de ordem. Com a comutação de pacotes, não há nenhum caminho, e assim diferentes pacotes podem seguir caminhos distintos, dependendo das condições da rede no momento em que eles são enviados. Portanto, eles podem chegar fora de ordem.

A comutação de pacotes é mais tolerante a defeitos que a comutação de circuitos. De fato, foi esse o motivo de sua criação. Se um switch ficar inativo, todos os circuitos que o utilizam serão encerrados, e nenhum tráfego poderá mais ser transmitido em qualquer deles. Com a comutação de pacotes, os pacotes poderão ser roteados de modo a contornar switches inativos.

A configuração de um caminho com antecedência também abre a possibilidade de

se reservar largura de banda com antecedência. Se a largura de banda for reservada, quando um pacote chegar, ele poderá ser transmitido de imediato sobre a largura de banda reservada. Com a comutação de pacotes, nenhuma largura de banda é reservada, e assim talvez os pacotes tenham de esperar sua vez para serem encaminhados.

Reservar largura de banda com antecedência significa que não poderá ocorrer nenhum congestionamento quando surgir um pacote (a menos que surja um número de pacotes maior que o esperado). Por outro lado, quando for feita uma tentativa de estabelecer um circuito, a tentativa poderá falhar devido ao congestionamento. Desse modo, o congestionamento poderá ocorrer em momentos diferentes com a comutação de circuitos (em tempo de configuração) e com a comutação de pacotes (quando os pacotes forem enviados).

Se um circuito tiver sido reservado para um determinado usuário e não houver tráfego para enviar, a largura de banda desse circuito será desperdiçada. Ele não poderá ser usado em outro tráfego. A comutação de pacotes não desperdiça largura de banda e, portanto, é mis eficiente do ponto de vista do sistema como um todo. É crucial entender esse compromisso para se compreender a diferença entre comutação de circuitos e comutação de pacotes. O compromisso se dá entre serviço garantido e desperdício de recursos *versus* serviço não garantido sem desperdício de recursos.

A comutação de pacotes utiliza a transmissão store-and-forward. Um pacote é acumulado na memória de um roteador, e depois é enviado ao roteador seguinte. Com a comutação de circuitos, os bits simplesmente fluem de forma contínua pelo fio. A técnica store-and-forward aumenta o retardo.

Outra diferença é que a comutação de circuitos é completamente transparente. O transmissor e o receptor podem usar qualquer taxa de bits, formato ou método de enquadramento que desejarem. A concessionária de comunicações não toma

conhecimento dessas informações. Com a comutação de pacotes, a concessionária de comunicações determina os parâmetros básicos. Grosso modo, para mostrar as diferenças entre essas tecnologias, poderíamos comparar uma rodovia a uma estrada de ferro. Na primeira, o usuário determina o tamanho, a velocidade e o tipo de veículo; na outra, a concessionária de comunicações trata de todos esses detalhes. É essa transparência que permite a coexistência de voz, dados e mensagens de fax no sistema telefônico.

Uma última diferença entre a comutação de circuitos e a de pacotes é o algoritmo de tarifação. Com a comutação de circuitos, a tarifação se baseava historicamente na distância e no tempo. No caso dos telefones móveis, em geral a distância não é importante, exceto para chamadas internacionais, e o tempo desempenha apenas um papel secundário (por exemplo, um plano de chamadas com 2000 minutos gratuitos custa mais que um plano com 1000 minutos gratuitos e, algumas vezes, chamadas noturnas ou nos finais de semana são mais econômicas que o normal. Com a comutação de pacotes, o tempo de conexão não é importante, mas o volume do tráfego às vezes tem importância. Para usuários domésticos, em geral os provedores cobram uma tarifa fixa mensal, porque é menos trabalhoso para eles e mais fácil de entender para os clientes, mas as concessionárias de backbones cobram das redes regionais com base no volume de seu tráfego. As diferenças estão resumidas na Figura 2.40.

[arte: ver original p. 151]

[T]Tabela

Item	**Comutação de circuitos**	**Comutação de pacotes**
Configuração de chamadas	Obrigatória	Não necessária
Caminho físico dedicado	Sim	Não
Cada pacote segue a mesma rota	Sim	Não
Os pacotes chegam em ordem	Sim	Não

A falha de um switch é fatal Sim Não

Largura de banda disponível Fixa Dinâmica

Momento de possível congestionamento Durante a configuração Em todos os pacotes

Largura de banda potencialmente desperdiçada Sim Não

Transmissão store-and-forward Não Sim

Transparência Sim Não

Tarifação Por minuto Por pacote

[F]Figura 2.40

[FL] Uma comparação entre redes de comutação de circuitos e redes de comutação de pacotes

A comutação de circuitos e a comutação de pacotes são tão importantes que voltaremos a elas em breve e descreveremos em detalhes as diversas tecnologias utilizadas.

[T2] 2.6 O sistema de telefonia móvel

O sistema telefônico tradicional (ainda que ele algum dia chegue a vários gigabits entre uma extremidade e outra da fibra) não será capaz de satisfazer a um grupo crescente de usuários: as pessoas em trânsito. Agora, as pessoas esperam efetuar chamadas telefônicas de aviões, carros, piscinas e enquanto fazem jogging no parque. Dentro de alguns anos, elas também irão querer enviar correio eletrônico e navegar na Web enquanto estiverem em todos esses lugares e em muitos outros. Conseqüentemente, há um enorme interesse na telefonia sem fios. Nas seções seguintes, estudaremos esse tópico em detalhes.

Há duas variedades básicas de telefones sem fios: os telefones sem fios propriamente ditos e os telefones móveis (às vezes chamados **telefones**

celulares). Os **telefones sem fios** são dispositivos que consistem em uma estação básica e um fone (ou aparelho) vendidos em conjunto para uso dentro de casa. Esses aparelhos nunca são usados na interligação de redes, e assim não os examinaremos mais neste livro. Em vez disso, vamos nos concentrar no sistema móvel, utilizado na comunicação de voz e dados em áreas extensas.

Os **telefones móveis** passaram por três gerações distintas, com diferentes tecnologias:

1. Voz analógica.
2. Voz digital.
3. Voz digital e dados (Internet, correio eletrônico etc.).

Embora a maior parte de nossa discussão seja sobre a tecnologia desses sistemas, é interessante observar como pequenas decisões políticas e de marketing podem ter um enorme impacto. O primeiro sistema móvel foi criado nos Estados Unidos pela AT&T e regulamentado para todo o país pela FCC. Como resultado, o território inteiro dos EUA tinha um único sistema (analógico), e um telefone móvel adquirido na Califórnia também funcionava em Nova York. Em contraste, quando a tecnologia chegou à Europa, cada país criou seu próprio sistema, o que resultou em um fiasco.

A Europa aprendeu com seus erros e, ao surgir a tecnologia digital, as PTTs estatais se juntaram e padronizaram um único sistema (GSM); portanto, qualquer telefone móvel europeu funcionará em qualquer lugar da Europa. Na época, os EUA haviam decidido que o governo não deveria participar do esforço de padronização, e assim a padronização da tecnologia digital ficou a cargo do mercado. Essa decisão resultou em diferentes fabricantes de equipamentos que produziram tipos distintos de telefones móveis. Em conseqüência disso, os Estados Unidos agora têm dois importantes sistemas de telefonia móvel digital incompatíveis em operação (além de mais um sistema secundário).

Apesar da liderança inicial dos EUA, a propriedade e a utilização da telefonia móvel na Europa é agora muito maior que nos Estados Unidos. O fato de haver um único sistema para toda a Europa explica em parte esse fato, mas há outras razões. Um segundo ponto em que os EUA e a Europa divergiram foi a questão dos números de telefone. Nos EUA, os telefones móveis têm números misturados com os telefones comuns (fixos). Desse modo, um chamador não tem como saber se, digamos, (212) 234-5678 é um telefone fixo (com uma ligação de baixo custo ou gratuita) ou um telefone móvel (com uma tarifa cara). Para impedir que as pessoas ficassem receosas de usar o telefone, as empresas de telefonia decidiram fazer o proprietário do telefone móvel pagar pelas chamadas recebidas. Em conseqüência disso, muitas pessoas hesitaram em comprar um telefone móvel por medo de terem de pagar uma conta enorme apenas por receberem ligações. Na Europa, os telefones móveis têm um código de área especial (análogo aos números 800 e 900) e assim podem ser reconhecidos instantaneamente. Como resultado, a regra habitual de fazer o chamador pagar também se aplica aos telefones móveis da Europa (com exceção das ligações internacionais, cujos custos são divididos).

Uma terceira questão que teve grande impacto na adoção da telefonia móvel é o uso difundido de telefones pré-pagos na Europa (até 75% em algumas regiões). Esses telefones podem ser adquiridos em muitas lojas sem mais formalidades que a compra de um aparelho de rádio. Você paga e leva. Eles são pré-carregados com, por exemplo, 20 ou 50 euros em ligações e podem ser recarregados (com a utilização de um código PIN secreto) quando o saldo cai até zero. Por essa razão, praticamente todos os adolescentes e muitas crianças na Europa têm telefones móveis (em geral pré-pagos) para que seus pais possam localizá-los, sem o perigo de terem de pagar uma conta enorme. Se o telefone móvel for usado apenas ocasionalmente, seu uso será quase gratuito, pois não haverá tarifa

mensal nem por chamadas recebidas.

[T3] 2.6.1 Telefones móveis de primeira geração: voz analógica

Já estudamos os aspectos políticos e de marketing dos telefones móveis. Agora, vamos examinar a tecnologia, começando pelo sistema mais antigo. Os radiotelefones móveis eram usados esporadicamente na comunicação militar e marítima, durante as primeiras décadas do Século XX. Em 1946, foi criado em St. Louis, nos EUA, o primeiro sistema para telefones baseados em automóveis. O sistema utilizava um único transmissor grande no topo de um alto edifício e tinha um único canal, usado para transmissões e recepções. Para conversar, o usuário tinha de apertar um botão que ativava o transmissor e desativava o receptor. Tais sistemas, conhecidos como **sistemas "push-to-talk"**, foram instalados em diversas cidades a partir dos anos 50. Sistemas de radioamador, táxis e carros da polícia nos programas de televisão utilizavam com freqüência essa tecnologia.

Na década de 1960, o **IMTS (Improved Mobile Telephone System — sistema de telefonia móvel aperfeiçoado)** foi instalado. Ele também utilizava um transmissor de alta potência (200 watts) no topo de uma montanha, mas agora tinha duas freqüências, uma para transmissão e outra para recepção. Por isso, o botão "apertar para falar" (push-to-talk) não era mais necessário. Tendo em vista que toda a comunicação dos telefones móveis utilizava um canal para transmissão e outro para recepção dos sinais, os usuários móveis não podiam ouvir uns aos outros (ao contrário do que acontecia com o sistema utilziado nos táxis).

O IMTS admitia 23 canais espalhados pelas freqüências de 150 a 450 MHz. Devido ao pequeno número de canais, muitas vezes os usuários tinham de esperar muito tempo antes de obter um tom de discagem. Além disso, devido à alta potência do transmissor, os sistemas adjacentes tinham de estar a diversos quilômetros de distância uns dos outros para evitar interferência. Em suma, o

sistema era impraticável devido à sua limitada capacidade.

[T4] AMPS (Advanced Mobile Phone System)

Tudo isso mudou com o **AMPS (Advanced Mobile Phone System — sistema avançado de telefonia móvel)**, inventado pelo Bell Labs e que foi instalado primeiramente nos Estados Unidos em 1982. Ele também foi usado na Inglaterra, onde recebeu o nome TACS, e no Japão, onde foi chamado MCS-L1. Embora não seja mais o estado da arte, vamos examiná-lo com alguma profundidade, porque muitas de suas propriedades fundamentais foram herdadas diretamente por seu sucessor digital, o D-AMPS, a fim de conseguir compatibilidade retroativa.

Em todos os sistemas de telefonia móvel, uma região geográfica é dividida em **células**, e é esse o motivo pelo qual esses dispositivos são chamados às vezes telefones celulares. No AMPS, as células têm em geral 10 a 20 km; nos sistemas digitais, as células são menores. Cada célula utiliza algum conjunto de freqüências não utilizado por qualquer das células vizinhas. A idéia fundamental que dá aos sistemas celulares uma capacidade muito maior que a dos sistemas anteriores é o uso de células relativamente pequenas e a reutilização de freqüências de transmissão em células vizinhas (mas não adjacentes). Enquanto um sistema IMTS com um alcance de 100 km pode ter uma chamada em cada freqüência, um sistema AMPS pode ter 100 células de 10 km na mesma região e é capaz de estabelecer de 5 a 10 chamadas em cada freqüência, em células amplamente separadas. Portanto, a estrutura celular aumenta a capacidade do sistema em pelo menos uma ordem de grandeza, ou mais se as células forem menores. Além disso, células menores significam menor necessidade de energia, o que possibilita a existência de dispositivos transmissores e receptores menores e mais econômicos. Os telefones portáteis consomem 0,6 watt; os transmissores de carro geralmente necessitam de 3 watts, o máximo permitido pela FCC.

A idéia de reutilização de freqüências é ilustrada na Figura 2.41(a). Em geral, as células são razoavelmente circulares; porém, é mais simples representá-las como hexágonos. Na Figura 2.41(a), todas as células têm o mesmo tamanho. As células são agrupadas em unidades de sete células. Cada letra indica um grupo de freqüências. Observe que, para cada conjunto de freqüências, existe um buffer de aproximadamente duas células de extensão, no qual essa freqüência não é reutilizada, o que proporciona boa separação e pouca interferência.

Encontrar locais altos para colocar antenas de estação base é uma questão fundamental. Esse problema levou algumas concessionárias de telecomunicações a fazer alianças com a Igreja Católica Romana, que possui um número significativo de locais potenciais para a instalação de antenas em todo o mundo, todas convenientemente controladas por uma única entidade.

Em uma área em que o número de usuários cresce a ponto do sistema ficar sobrecarregado, a potência é reduzida, e as células sobrecarregadas são divididas em células menores chamadas **microcélulas** para permitir maior reutilização de freqüências, como mostra a Figura 2.41(b). Algumas vezes, as empresas de telefonia criam microcélulas temporárias, utilizando torres portáteis com enlaces de satélite para atender à demanda de eventos esportivos, consertos de rock e outros eventos em que um grande número de usuários de telefones celulares se reúnem por algumas horas. O tamanho que essas células devem ter é uma questão complexa, tratada em (Hac, 1995).

[arte: ver original p. 155]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 2.41

[FL] (a) As freqüências não são reutilizadas nas células adjacentes. (b) Para

acrescentar usuários, podem ser utilizadas células menores

No centro de cada célula há uma estação base que recebe as transmissões de todos os telefones presentes na célula. A estação base consiste em um computador e um transmissor/receptor conectados a uma antena. Em um sistema de pequeno porte, todas as estações base estão conectadas a um único dispositivo chamado **MTSO (Mobile Telephone Switching Oflice — estação de comutação de telefonia móvel)** ou **MSC (Mobile Switching Center — centro de comutação móvel).** Em um sistema maior, podem ser necessárias diversas MTSOs, todas conectadas a uma MTSO de segundo nível e assim por diante. Basicamente, as MTSOs são estações finais, como acontece no sistema telefônico. Na verdade, elas estão conectadas a pelo menos uma estação final de um sistema telefônico. As MTSOs se comunicam com as estações base, entre si e com a PSTN, usando uma rede de comutação de pacotes.

Em qualquer instante, cada telefone móvel ocupa logicamente uma célula específica e está sob o controle da estação base dessa célula. Quando um telefone móvel deixa fisicamente uma célula, sua estação base detecta que o sinal do telefone está se enfraquecendo e questiona todas as estações base vizinhas quanto à quantidade de energia que elas estão recebendo dele. Em seguida, a estação base faz a transferência para a célula que está obtendo o sinal mais forte, ou seja, a célula em que o telefone está localizado no momento. O telefone é então informado de quem é o seu novo chefe e, se houver uma chamada em andamento, ele será solicitado a passar para outro canal (porque o antigo não é reutilizado em nenhuma das células adjacentes). Esse processo é chamado **handoff** e leva cerca de 300 ms. A atribuição de canais é feita pela MTSO, o centro nervoso do sistema. As estações base são apenas retransmissões de rádio. Os handoffs pode ser feito de dois modos. Em um **soft handoff**, o sinal do

telefone é adquirido pela nova estação base antes da anterior se desconectar. Desse modo, não existe nenhuma perda de continuidade. A desvantagem aqui é que o telefone precisa ter a capacidade de sintonizar duas freqüências ao mesmo tempo (a antiga e a nova). Nem os dispositivos de primeira geração nem os de segunda geração podem fazer isso.

Em um **hard handoff**, a estação base antiga libera o sinal do telefone antes de ele ser adquirido pela nova estação base. Se a nova não for capaz de adquiri-lo (por exemplo, porque não existe nenhuma freqüência disponível), a chamada será desconectada de forma abrupta. Os usuários tendem a notar essa interrupção, mas ela ocasionalmente é inevitável com a estrutura atual.

[T4] Canais

O sistema AMPS utiliza 832 canais full-duplex, cada um consistindo em um par de canais simplex. Existem 832 canais de transmissão simplex de 824 a 849 MHz, e 832 canais de recepção simplex de 869 a 894 MHz. Cada um desses canais simplex tem 30 kHz de largura. Desse modo, o AMPS utiliza FDM para separar os canais.

Na faixa de 800 MHz, as ondas de rádio têm cerca de 40 cm de comprimento e trafegam em linha reta. Elas são absorvidas por árvores e plantas, e ricocheteiam no chão e nos prédios. É possível que o sinal enviado por um telefone móvel alcance a estação base pelo caminho direto, mas também pode chegar um pouco mais tarde, depois de ricochetear no chão ou em um prédio. Isso pode causar um efeito de eco ou de distorção no sinal (esmaecimento de vários caminhos). Às vezes, é possível até mesmo ouvir uma conversação distante que ecoou várias vezes.

Os 832 canais estão divididos em quatro categorias:

1. Controle (da base para a unidade móvel) para gerenciar o sistema.

2. Localização (da base para a unidade móvel) para alertar os usuários móveis de chamadas destinadas a eles.

3. Acesso (bidirecional) para configuração de chamadas e atribuição de canais.

4. Dados (bidirecional) para voz, fax ou dados.

Vinte e um desses canais são reservados para controle, e eles estão conectados a uma unidade de memória PROM em cada telefone. Como as mesmas freqüências não podem ser reutilizadas em células vizinhas, o número real de canais de voz disponíveis por célula é bem menor que 832; em geral, esse número é de aproximadamente 45.

[T4] Gerenciamento de chamadas

No AMPS, cada telefone móvel tem um número de série de 32 bits e um número de telefone de 10 dígitos em sua PROM. O número de telefone é representado como um código de área de 3 dígitos em 10 bits e um número de assinante de 7 dígitos em 24 bits. Quando é contactado, um telefone varre uma lista pré-programada de 21 canais de controle até encontrar o sinal mais forte.

Em seguida, o telefone transmite seu número de série de 32 bits e o número de telefone de 34 bits. A exemplo de todas as outras informações de controle do AMPS, esse pacote é enviado várias vezes em formato digital e com um código de correção de erros, apesar de os próprios canais de voz serem analógicos.

Quando ouve a mensagem, a estação base avisa à MTSO, que registra a existência de seu novo cliente e também informa a localização atual do cliente à sua MTSO inicial. Durante a operação normal, o telefone móvel repete o registro uma vez a cada 15 minutos, em média.

Para fazer uma chamada, um usuário móvel liga o telefone, digita no teclado o número a ser chamado e pressiona o botão SEND. Em seguida, o telefone transmite o número a ser chamado e sua própria identidade no canal de acesso.

Se houver uma colisão, ele tenta novamente mais tarde. Ao receber a solicitação, a estação base informa à MTSO. Se o chamador for um cliente da empresa da MTSO (ou de uma de suas parceiras), a MTSO procura um canal disponível para a chamada. Se o encontrar, o número do canal será enviado de volta no canal de controle. Em seguida, o telefone móvel se conecta automaticamente ao canal de voz selecionado e aguarda até que a parte chamada atenda ao telefone.

As chamadas recebidas funcionam de forma diferente. Para começar, todos os telefones inativos ouvem continuamente o canal de localização para detectar as mensagens destinadas a eles. Quando é feita uma chamada para um telefone móvel (a partir de um telefone fixo ou de outro telefone móvel), um pacote é enviado à MTSO local do telefone chamado, para que ele seja localizado. Em seguida, é enviado um pacote à estação base em sua célula atual, que então envia um pacote de difusão no canal de localização com o formato: "Unidade 14, você está aí?". O telefone chamado responde "Sim" no canal de acesso. Depois, a base transmite algo como: "Unidade 14, chamada para você no canal 3". Nesse momento, o telefone chamado se conecta ao canal 3 e começa a emitir sinais sonoros (ou a tocar alguma melodia que o proprietário do telefone ganhou como presente de aniversário).

[T3] 2.6.2 Telefones móveis de segunda geração: voz digital

A primeira geração de telefones celulares era analógica; a segunda geração era digital. Da mesma maneira que não havia nenhuma padronização mundial durante a primeira geração, também não havia nenhuma padronização durante a segunda. Quatro sistemas são usados agora: D-AMPS, GSM, CDMA e PDC. A seguir, descreveremos os três primeiros. O PDC só é usado no Japão e é basicamente o D-AMPS modificado para compatibilidade retroativa com o sistema analógico japonês de primeira geração. O nome **PCS (Personal Communications**

Services — serviços de comunicações pessoais) às vezes é usado na literatura de marketing para indicar um sistema de segunda geração (isto é, digital). Originalmente, ele representava um telefone celular que emprega a banda de 1900 MHz, mas essa distinção é rara nos dias de hoje.

[T4] D-AMPS (Digital Advanced Mobile Phone System)

A segunda geração dos sistemas AMPS é o **D-AMPS**, totalmente digital. Ele é descrito no padrão internacional IS-54 e em seu sucessor o IS-136. O D-AMPS foi cuidadosamente projetado para coexistir com o AMPS, de tal forma que os telefones móveis de primeira e de segunda geração pudessem operar simultaneamente na mesma célula. Em particular, o D-AMPS utiliza os mesmos canais de 30 kHz que o AMPS, e nas mesmas freqüências, de modo que um único canal possa ser analógico e os canais adjacentes possam ser digitais. Dependendo da mistura de telefones em uma célula, a MTSO define quais dos canais serão analógicos e quais serão digitais, e pode mudar os tipos de canais dinamicamente, à medida que se altera a mistura de telefones em uma célula. Quando o D-AMPS foi lançado como um serviço, uma nova banda de freqüência se tornou disponível para lidar com o aumento esperado da carga. Os canais upstream estavam na faixa de 1850 a 1910 MHz, e os canais downstream correspondentes estavam na faixa de 1930 a 1990 MHz, novamente aos pares, como no AMPS. Nessa banda, as ondas têm 16 cm de comprimento, e assim uma antena padrão de ¼ de onda tem apenas 4 cm de comprimento, lidando com telefones menores. Porém, muitos telefones D-AMPS podem usar as bandas de 850 MHz e 1900 MHz, a fim de obter uma faixa mais ampla de canais disponíveis. Em um telefone móvel D-AMPS, o sinal de voz captado pelo microfone é digitalizado e compactado com o uso de um modelo mais sofisticado que os esquemas de modulação delta e codificação profética que estudamos antes. A

compactação leva em conta propriedades detalhadas do sistema vocal humano para levar a largura de banda dos 56 kbps padrão da codificação PCM para 8 kbps ou menos. A compactação é feita por um circuito chamado **vocoder** (Bellamy, 2000). A compactação é realizada no telefone, e não na estação base ou na estação final, a fim de reduzir o número de bits transmitidos pelo enlace aéreo. Com a telefonia fixa, não há vantagem em fazer a compactação no telefone, pois a redução do tráfego no loop local não aumenta de forma alguma a capacidade do sistema.

No caso da telefonia móvel existe uma enorme vantagem na realização da digitalização e da compactação no aparelho telefônico tanto que, no D-AMPS, três usuários podem compartilhar um único par de freqüências usando multiplexação por divisão de tempo. Cada para de freqüências admite 25 quadros/s de 40 ms cada quadro. Cada quadro se divide em seis slots de tempo de 6,67 ms cada um, como ilustra a Figura 2.42(a) para o par de freqüências mais baixo.

[arte: ver original p. 158]

[Dísticos]

[1] Quadro TDM Quadro TDM

40 ms 40 ms

[2]

Upstream 1 2 3 1 2 3 1850,01 MHz 1 2 3 4 5 6 1850,01 MHz

Unidade móvel para base Unidade móvel para base

[3]

Downstream 3 1 2 3 1 2 1930,05 MHz 6 1 2 3 4 5 1930,05 MHz

Base para unidade móvel Base

para unidade móvel

[4] Slot de 324 bits:

64 bits de controle

101 bits de correção de erros

159 bits de dados de voz

(a) (b)

[F]Figura 2.42

[FL] (a) Um canal D-AMPS com três usuários. (b) Um canal D-AMPS com seis usuários

Cada quadro contém três usuários que utilizam os enlaces upstream e downstream em turnos. Por exemplo, durante o slot 1 da Figura 2.42(a), o usuário 1 pode transmitir para a estação base e o usuário 3 está recebendo a transmissão da estação base. Cada slot tem 324 bits de comprimento, dos quais 64 bits são utilizados para finalidades de tempos de proteção, sincronização e controle, restando 260 bits para carga útil do usuário. Dos bits de carga útil, 101 são usados para correção de erros no ruidoso enlace de ar, e assim só restam de fato 159 bits para voz compactada. Com 50 slots/s, a largura de banda disponível para voz compactada está abaixo de 8 kbps, 1/7 da largura de banda padrão do PCM.

Utilizando-se melhores algoritmos de compactação, é possível reduzir a voz a 4 kbps e, nesse caso, seis usuários podem ser acomodados em um quadro, como ilustra a Figura 2.42(b). Da perspectiva da operadora, a capacidade de forçar o aumento de três para seis usuários do D-AMPS no mesmo espectro que um único usuário AMPS é um enorme ganho e explica grande parte da popularidade do PCS. É claro que a qualidade de voz a 4 kbps não é comparável ao que pode ser obtido a 56 kbps, mas poucas operadoras PCS anunciam sua qualidade de som

como alta fidelidade. Também deve ficar claro que, no caso de dados, um canal de 8 kbps não tem nem mesmo a qualidade de um antigo modem de 9600 bps.
A estrutura de controle do D-AMPS é bastante complicada. Em resumo, grupos de 16 quadros formam um superquadro, com certas informações de controle presentes em cada superquadro um número limitado de vezes. Seis canais de controle principais são usados: configuração de sistema, controle em tempo real e não de tempo real, localização, resposta a acesso e mensagens curtas. Porém, conceitualmente, ele funciona como o AMPS. Quando um telefone móvel é ligado, ele faz contato com a estação base para se anunciar, e depois escuta um canal de controle em busca de chamadas recebidas. Tendo captado uma nova unidade móvel, a MTSO informa à base inicial do usuário onde ele está, de forma que as chamadas possam ser roteadas corretamente.
Uma diferença entre o AMPS e o D-AMPS é a forma como o handoff é tratado. No AMPS, a MTSO o administra completamente, sem ajuda dos dispositivos móveis. Como podemos observar na Figura 2.42, no D-AMPS, durante 1/3 do tempo uma unidade móvel não está transmitindo nem recebendo. Ela utiliza esses slots ociosos para medir a qualidade da linha. Ao descobrir que o sinal está diminuindo, ela reclama à MTSO, que pode então interromper a conexão; nesse momento, a unidade móvel pode tentar sintonizar um sinal mais forte de outra estação base. Como no AMPS, ela ainda demora cerca de 300 ms para efetuar o handoff. Essa técnica é chamada **MAHO (Mobile Assistente HandOff — handoff com participação da unidade móvel).**

[T4] GSM (Global System for Mobile Communications)

O D-AMPS é amplamente utilizado nos Estados Unidos e (em forma modificada) no Japão. Em quase todos os lugares do mundo é utilizado um sistema chamado **GSM (Global System for Mobile Communications — sistema global para**

comunicações móveis), e ele está começando a ser usado até mesmo nos EUA, em escala limitada. Em uma primeira aproximação, o GSM é semelhante ao D-AMPS. Ambos são sistemas celulares. Nos dois sistemas, é empregada a multiplexação por divisão de freqüência, com cada unidade móvel transmitindo em uma freqüência e recebendo em uma freqüência mais alta (80 MHz mais alta no caso do D-AMPS, 55 MHz mais alta no caso do GSM). Também em ambos os sistemas, um único par de freqüências é dividido pela multiplexação por divisão de tempo em slots (períodos) de tempo compartilhados por várias unidades móveis. Porém, os canais GSM são muito mais largos que os canais AMPS (200 kHz *versus* 30 kHz) e contêm relativamente poucos usuários adicionais (8 *versus* 3), dando ao GSM uma taxa de dados muito maior por usuário que o D-AMPS. Descreveremos a seguir algumas das principais propriedades do GSM. Entretanto, o padrão GSM impresso tem mais de 5000 [*sic*] páginas. Uma grande fração desse material se relaciona aos aspectos de engenharia do sistema, em especial ao projeto dos receptores para tratar a propagação de sinais por vários caminhos, e à sincronização de transmissores e receptores. Nada disso será sequer mencionado a seguir.

Cada banda de freqüência tem 200 kHz de largura, como mostra a Figura 2.43. Um sistema GSM tem 124 pares de canais simplex. Cada canal simplex tem 200 kHz de largura e admite oito conexões separadas, usando a multiplexação por divisão de tempo. Cada estação atualmente ativa recebe a atribuição de um slot de tempo em um par de canais. Teoricamente, 992 canais podem ser admitidos em cada célula, mas muitos deles não estão disponíveis, a fim de evitar conflitos de freqüências com as células vizinhas. Na Figura 2.43, todos os oito slots de tempo sombreados pertencem à mesma conexão, quatro deles em cada sentido. A transmissão e a recepção não acontecem no mesmo slot de tempo, porque os rádios GSM não podem transmitir e receber ao mesmo tempo, e a passagem de

um sentido para o outro demora algum tempo. Se a unidade móvel atribuída à faixa de 890,4/935,4 MHz e ao slot de tempo 2 quisesse transmitir algo para a estação base, ela usaria os quatro slots sombreados inferiores (e os slots seguintes a eles no tempo), inserindo alguns dados em cada slot até todos os dados terem sido enviados.

[arte: ver original p. 160]

[Dísticos]

[1]Freqüência

[2]Quadro TDM Canal

959,8 MHz 124

[3] Base para unidade móvel

935,4 MHz 2

935,2 MHz 1

[4]914,8 MHz 124

[5] Unidade móvel para base

890,4 MHz 2

890,2 MHz 1

Tempo

[F]Figura 2.43

[FL] O GSM utiliza 124 canais de freqüência, cada um dos quais emprega um sistema TDM de oito slots

Os slots TDM mostrados na Figura 2.43 fazem parte de uma complexa hierarquia de enquadramento. Cada slot TDM tem uma estrutura específica, e grupos de slots TDM formam multiquadros, também com uma estrutura específica. Uma versão simplificada dessa hierarquia é mostrada na Figura 2.44. Aqui, podemos ver que cada slot TDM consiste em um quadro de dados de 148 bits que ocupa o

canal por 577 μs (incluindo um tempo de proteção de 30 μs depois de cada slot). Cada quadro de dados começa e termina com três bits 0, para fins de delineação de quadros. Ele também contém dois campos *Information* de 57 bits, cada um com um bit de controle que indica se o campo *Information* seguinte se refere a voz ou dados. Entre os campos *Information* há um campo (de treinamento) *Sync* de 26 bits, usado pelo receptor para realizar a sincronização até os limites de quadro do transmissor.

[arte: ver original p. 161]

[Dísticos]

[1] Multiquadro de 32.500 bits enviado em 120 ms

0 1 2 3 4 5 6 7 8 9 10 11 CTL 13 14 15 16 17 18 19 20 21 22 23 24

[2] Reservado para uso futuro

[3] Quadro TDM de 1250 bits enviado em 4,615 ms

0 1 2 3 4 5 6 7

[4] Tempo de proteção de 8,25 bits (30 μs)

[5] Quadro de dados de 148 bits enviado em 547 μs

000 Information Sync Information 000

Bits 3 57 26 57 3

Bit de voz/dados

[F]Figura 2.44

[FL] Uma parte da estrutura de enquadramento GSM

Um quadro de dados é transmitido em 547 μs, mas um transmissor só pode enviar um quadro de dados a 4,615 ms, pois ele está compartilhando o canal com sete outras estações. A taxa bruta de cada canal é 270.833 bps, dividida entre oito usuários. Isso fornece 33,854 kbps brutos, mais que o dobro dos 324 bits transmitidos 50 vezes por segundo para 16,2 kbps no D-AMPS. Porém, como

ocorre com o AMPS, o overhead consome uma grande fração da largura de banda, deixando em última análise 24,7 kbps de carga útil por usuário antes da correção de erros. Após a correção de erros, restam 13 kbps para voz, oferecendo uma qualidade de voz bem melhor que a do D-AMPS (ao custo de usar mais largura de banda).

Como podemos ver na Figura 2.44, oito quadros de dados formam um quadro TDM, e 26 quadros TDM formam um multiquadro de 120 ms. Dos 26 quadros TDM em um multiquadro, o slot 12 é usado para controle e o slot 25 é reservado para uso futuro, e assim somente 24 estão disponíveis para tráfego do usuário. Porém, além do multiquadro de 26 slots mostrado na Figura 2.44, também é usado um multiquadro de 51 slots (não mostrado). Alguns desses slots são empregados para guardar diversos canais de controle usados para gerenciar o sistema. O **canal de controle de difusão** é um fluxo contínuo de saída da estação base, contendo a identidade da estação base e o status do canal. Todas as estações móveis monitoram a intensidade de seu sinal para verificar quando elas são transferidas para uma nova célula.

O **canal de controle dedicado** é usado para atualização de local, registro e configuração de chamadas. Em particular, cada estação base mantém um banco de dados das estações móveis que estão atualmente sob sua jurisdição. As informações necessárias para manter esse banco de dados são enviadas sobre o canal de controle dedicado.

Por fim, existe o **canal de controle comum**, dividido em três subcanais lógicos. O primeiro deles é o **canal de localização**, que a estação base utiliza para anunciar as chamadas recebidas. Cada estação móvel monitora continuamente esse canal para verificar se há chamadas que ela deva responder. O segundo é o **canal de acesso aleatório**, que permite aos usuários solicitarem um slot no canal de controle dedicado. Se duas solicitações colidirem, elas serão adulteradas e terão

de ser repetidas mais tarde. Usando o slot do canal de controle dedicado, a estação pode configurar uma chamada. O slot atribuído é anunciado no terceiro subcanal, o **canal de concessão de acesso.**

[T4] CDMA (Code Division Multiple Access)

O D-AMPS e o GSM são sistemas bastante convencionais. Ambos utilizam a FDM e a TDM para dividir o espectro em canais, e os canais em slots de tempo. Porém, existe um terceiro tipo de sistema, o **CDMA (Code Division Multiple Access — acesso múltiplo por divisão de código)**, que funciona de modo completamente diferente. Quando o CDMA foi proposto pela primeira vez, a indústria o recebeu quase com a mesma reação que a Rainha Isabel teve quando Colombo propôs alcançar a Índia navegando na direção oposta. Porém, graças à persistência de uma única empresa, a Qualcomm, o CDMA amadureceu até ser considerado não apenas aceitável, mas ser visto hoje como a melhor solução técnica existente, e também como a base dos sistemas móveis de terceira geração. Ele é amplamente utilizado nos EUA em sistemas móveis de segunda geração, competindo de frente com o D-AMPS. Por exemplo, o PCS da Sprint utiliza CDMA, enquanto o Wireless da AT&T usa o D-AMPS. O CDMA é descrito no padrão internacional IS-95 e às vezes é referido por esse nome. Também é usado o nome de marca **cdmaOne.**

O CDMA é muito diferente do AMPS, do D-AMPS e do GSM. Em vez de dividir a faixa de freqüências permitida em algumas centenas de canais estreitos, o CDMA permite que cada estação transmita sobre todo o espectro de freqüências durante todo o tempo. Várias transmissões simultâneas são separadas com o uso da teoria de codificação. O CDMA também relaxa a suposição de que quadros que colidem são totalmente adulterados. Em vez disso, ele pressupõe que vários sinais se somam de modo linear.

Antes de entrarmos no algoritmo, vamos considerar uma analogia: o saguão de

um aeroporto com muitos pares de pessoas conversando. Com a TDM, todas as pessoas estariam no meio do saguão, mas conversariam por turnos, um par de pessoas de cada vez. Com a FDM, as pessoas formariam grupos bem separados, cada grupo mantendo sua própria conversação ao mesmo tempo, mas ainda independente dos outros grupos. Com o CDMA, todas as pessoas estariam no meio do saguão falando ao mesmo tempo, mas cada par de pessoas conversaria em um idioma diferente. O par que estivesse falando em francês só reconheceria esse idioma, rejeitando tudo que não fosse francês como ruído. Desse modo, a chave para o CDMA é a capacidade de extrair o sinal desejado e rejeitar todos os outros como ruído aleatório. Veremos a seguir uma descrição um pouco simplificada do CDMA.

No CDMA, cada tempo de duração de um bit é subdividido em *m* intervalos curtos, denominados **chips.** Normalmente, existem 64 ou 128 chips por bit mas, no exemplo apresentado a seguir, usaremos 8 chips/bit por simplicidade.

A cada estação é atribuído um código de *m* bits exclusivo chamado **seqüência de chips.** Para transmitir um bit 1, cada estação envia sua seqüência de chips. Para transmitir um bit 0, envia o complemento de 1 dessa seqüência. Não são permitidos quaisquer outros padrões. Assim, para $m = 8$, se a estação *A* receber a atribuição da seqüência de chips 00011011, ela transmitirá um bit 1 ao enviar 00011011, e um bit 0 ao enviar 11100100.

O aumento do volume de informações a serem enviadas de *b* bits/s para *mb* chips/s só poderá ocorrer se a largura de banda disponível for aumentada *m* vezes, tornando o CDMA uma espécie de comunicação por espectro de dispersão (supondo-se a ausência de alterações nas técnicas de modulação ou de codificação). Se tivéssemos uma banda de 1 MHz disponível para 100 estações com FDM, cada uma teria 10 kHz e poderia transmitir a uma velocidade de 10 kbps (supondo-se 1 bit por Hz). No CDMA, cada estação utiliza 1 MHz inteiro e,

portanto, a taxa de chips é de 1 megachip por segundo. Com menos de 100 chips por bit, a largura de banda efetiva por estação é maior para o CDMA que para a FDM, e o problema de alocação de canais também é resolvido.

Para fins pedagógicos, é mais conveniente usar uma notação bipolar, com o 0 binário sendo –1 e o 1 binário sendo +1. Mostraremos as seqüências de chips entre parênteses; portanto, um bit 1 para a estação *A* agora se torna (–1–1–1+1+1–1+1+1). Na Figura 2.45(a), mostramos as seqüências de chips binárias atribuídas a quatro exemplos de estações. Na Figura 2.45(b), mostramos essas seqüências de chips em nossa notação bipolar.

[arte: ver original p. 163]

[Dísticos]

[1]	A: 00011011	A: (–1–1–1+1+1–1+1+1)
	B: 00101110	B: (–1–1+1–1+1+1+1–1)
	C: 01011100	C: (–1+1–1+1+1+1–1–1)
	D: 01000010	D: (–1 +1–1–1–1–1+1–1)
	(a)	(b)

[2]Seis exemplos:

––1–	C	S_1=(–1+1–1+1+1+1–1–1)
–11–	B + C	S_2=(–2 0 0 0+2+2 0–2)
10––	A + [ver símbolo]	S_3=(0 0–2+2 0–2 0+2)
101–	A + [ver símbolo] + C	S_4=(–1+1–3+3–1–1–1+1)
1111	A + B + C + D	S_5=(–4 0–2 0+2 0+2–2)
1101	A + B + [ver símbolo] + D	S_6=(–2–2 0–2 0–2+4 0)

(c)

[3]

$S_1 \bullet C$ = (1+1+1+1+1+1+1+1)/8= 1

$S_2 \bullet C$ = (2+0+0+0+2+2+0+2)/8= 1

S_3 • C = (0+0+2+2+0 −2+0−2)/8=0

S_4 • C = (1+1+3+3+1−1+1−1)/8= 1

S_5 • C = (4+0+2+0+2+0−2+2)/8=1

S_6 • C = (2−2+0−2+0−2−4+0)/8=−1

(d)

[F]Figura 2.45

[FL] (a) Seqüências de chips binárias para quatro estações (b) Seqüências de chips bipolares. (c) Seis exemplos de transmissões. (d) Recuperação do sinal da estação C

Cada estação tem sua própria seqüência exclusiva de chips. Vamos usar o símbolo **S** para indicar o vetor de *m* chips correspondente à estação *S*, e [ver símbolo] para sua negação. Todas as seqüências de chips são **ortogonais** par a par; isso significa que o produto interno normalizado de duas seqüências de chips distintas, **S** e **T** (indicado como **S • T**) é 0. Sabemos como gerar tal seqüência ortogonal de chips usando um método conhecido como **códigos de Walsh.** Em termos matemáticos, a ortogonalidade das seqüências de chips pode ser expressa por:

[Inserir equação do O.A. p. 164a] (4-5)

Em linguagem comum, o número de pares iguais é igual ao de pares diferentes. Essa propriedade da ortogonalidade será essencial mais adiante. Observe que, se **S • T** = 0, então [ver símbolo] também será 0. O produto interno normalizado de qualquer seqüência de chips por ela mesma é igual a 1:

[Inserir equação do O.A. p. 164b]

Isso ocorre porque cada um dos *m* termos do produto interno é 1 e, portanto, a soma é *m.* Observe também que [ver símbolo] = −1.

Durante cada intervalo com duração de um bit, uma estação pode transmitir um

bit 1 ao enviar sua seqüência de chips. Ela pode transmitir um bit 0 enviando a parte negativa de sua seqüência de chips, ou pode ficar inativa e não realizar qualquer transmissão. Por enquanto, supomos que todas as estações estão sincronizadas e que a transmissão de todas as seqüências de chips começa no mesmo instante.

Quando duas ou mais estações transmitem simultaneamente, seus sinais bipolares se somam linearmente. Por exemplo, se durante um período de um chip três estações transmitirem +1 e uma estação transmitir −1, o resultado será +2. Isso pode ser considerado como a soma de voltagens: se três estações transmitirem + 1 volt e uma estação transmitir −1 volt como saída, o resultado será 2 volts.

A Figura 2.45(c) apresenta seis exemplos em que uma ou mais estações transmitem ao mesmo tempo. No primeiro exemplo, *C* transmite um bit 1 e, assim, simplesmente obtemos a seqüência de chips de *C*. No segundo, *B* e *C* transmitem bits 1 e obtemos a soma de suas seqüências bipolares de chips, da seguinte forma:

(−1 -1 +1 -1 +1 +1 +1 −1) + (−1 +1 -1 +1 +1 +1 -1 −1) = (−2 0 0 0 +2 +2 0 −2)

No terceiro exemplo, a estação *A* envia um bit 1 e a estação *B* envia um bit 0. As outras permanecem inativas. No quarto exemplo, *A* e *C* enviam um bit 1, enquanto *B* envia um bit 0. No quinto exemplo, todas as quatro estações enviam um bit 1. Por fim, no último exemplo, *A*, *B* e *D* enviam um bit 1, enquanto *C* envia um bit 0. Observe que cada uma das seis seqüências — de S_1 até S_6 — mostradas na Figura 2.45(c) representa apenas um intervalo de tempo de um bit.

Para recuperar o fluxo de bits de uma estação individual, o receptor precisa conhecer com antecedência a seqüência de chips da estação transmissora. Ele executa a recuperação calculando o produto interno normalizado da seqüência de

chips recebida (a soma linear de todas as estações que transmitiram) e da seqüência de chips da estação cujo fluxo de bits está tentando recuperar. Se a seqüência de chips recebida for **S** e o receptor estiver tentando ouvir uma estação cuja seqüência de chips é **C**, ele apenas calcula o produto interno normalizado, **S** • **C**.

Para entender por que esse procedimento funciona, imagine que as duas estações *A* e *C* transmitem um bit 1 ao mesmo tempo em que *B* transmite um bit 0. O receptor percebe a soma **S** = **A** + [ver símbolo] + **C** e calcula:

S • **C** = (**A** + [ver símbolo] + **C**) • **C** = **A** • **C** + [ver símbolo] • **C** + **C** • **C** = 0 + 0 + 1 = 1

Os dois primeiros termos desaparecem, porque todos os pares de seqüências de chips foram cuidadosamente escolhidos para serem ortogonais, como mostra a Equação (2-4). Agora já deve estar claro por que essa propriedade precisa ser imposta às seqüências de chips.

Uma forma alternativa de analisar essa situação é imaginar que as três seqüências de chips chegaram todas separadas, em vez de somadas. Então, o receptor calcularia o produto interno de cada uma separadamente e somaria os resultados. Devido à propriedade de ortogonalidade, todos os produtos internos, exceto **C** • **C**, seriam iguais a 0. Somá-los e em seguida obter o produto interno é, na verdade, o mesmo que obter os produtos internos e depois somá-los.

Para tornar mais concreto o processo de decodificação, vamos considerar novamente os seis exemplos da Figura 2.45(c), ilustrados na Figura 2.45(d). Suponha que o receptor esteja interessado em extrair o bit enviado pela estação *C* de cada uma das seis somas de S_1 a S_6. Ele calcula o bit somando aos pares os produtos da **S** recebida com o vetor **C** da Figura 2.45(b), e depois extraindo 1/8 (pois $m = 8$, nesse caso). Como mostramos, o bit correto é decodificado de cada vez.

Em um sistema CDMA ideal sem ruídos, a capacidade (ou seja, o número de estações) pode se tornar arbitrariamente grande, do mesmo modo que a capacidade de um canal de Nyquist sem ruídos pode se tornar arbitrariamente grande, usando-se mais e mais bits por amostra. Na prática, as limitações físicas reduzem de forma considerável essa capacidade. Primeiro, supomos que todos os chips estejam sincronizados. Na realidade, tal sincronização é impossível. O que podemos fazer é sincronizar o transmissor e o receptor, obrigando o transmissor enviar uma seqüência de chips predefinida, longa o bastante para bloquear o receptor. Todas as outras transmissões (não sincronizadas) serão vistas então como ruído aleatório. Porém, mesmo que não haja um grande volume de ruído, o algoritmo básico de decodificação ainda funcionará perfeitamente bem. Existem vários resultados teóricos relacionando a superposição de seqüências de chips ao nível de ruído (Pickholtz *et al.*, 1982). Como se poderia esperar, quanto mais longa a seqüência de chips, maior a probabilidade de detectá-la corretamente na presença de um ruído. Para aumentar a confiabilidade, a seqüência de bits pode utilizar um código de correção de erros. As seqüências de chips nunca utilizam códigos de correção de erros.

Uma suposição implícita nessa abordagem é que os níveis de potência de todas as estações são iguais aos percebidos pelo receptor. O CDMA normalmente é usado em sistemas sem fios com uma estação base fixa e muitas estações móveis a distâncias variadas da estação base. Os níveis de potência recebidos na estação base dependem da distância em que se encontram os transmissores. Nesse caso, uma boa heurística é fazer cada estação móvel transmitir à estação base com a intensidade inversa do nível de potência que recebe da estação base. Assim, quando uma estação móvel receber um sinal fraco da estação base, ela utilizará maior potência do que uma estação móvel que receber um sinal forte. A estação base também pode instruir explicitamente as estações móveis a aumentar ou

diminuir sua potência de transmissão.

Supomos também que o receptor sabe quem é o transmissor. Em princípio, dada uma capacidade de computação suficiente, o receptor poderá escutar todos os transmissores ao mesmo tempo, executando o algoritmo de decodificação correspondente a cada um deles em paralelo. Na prática, é mais falar do que fazer. O CDMA também tem muitos outros fatores complicadores que não foram explicados nesta breve introdução. Apesar disso, o CDMA é um esquema inteligente que está sendo introduzido com rapidez para comunicação móvel sem fios. Normalmente, ele opera em uma faixa de 1,25 MHz (contra 30 kHz para o D-AMPS e 200 kHz para o GSM), mas admite muito mais usuários nessa faixa que qualquer um dos outros sistemas. Na prática, a largura de banda disponível para cada usuário é no mínimo tão boa quanto a do GSM e, com freqüência, muito melhor.

Os engenheiros que desejarem adquirir uma compreensão muito profunda do CDMA devem ler (Lee e Miller, 1998). Um esquema alternativo de dispersão, no qual a dispersão é realizada em relação ao tempo, e não em relação à freqüência, é descrito em (Crespo *et al.*, 1995). Outro esquema é descrito em (Sari *et al.*, 2000). Todas essas referências exigem um certo conhecimento de engenharia de comunicações.

[T3] 2.6.3 Telefones móveis de terceira geração: voz e dados digitais

Qual é o futuro da telefonia móvel? Vamos fazer uma breve análise. Diversos fatores estão orientando a indústria. Primeiro, o tráfego de dados já excede o tráfego de voz na rede fixa e está crescendo de forma exponencial, enquanto o tráfego de voz é essencialmente plano. Muitos especialistas da indústria esperam que, em breve, o tráfego de dados também domine o tráfego de voz em dispositivos móveis. Em segundo lugar, as indústrias de telefonia, entretenimento

e informática já se tornaram digitais e estão convergindo rapidamente. Muitas pessoas estão entusiasmadas com um dispositivo leve e portátil que atua como telefone, reprodutor de CDs, reprodutor de DVDs, terminal de correio eletrônico, interface da Web, máquina de jogos, processador de textos e muito mais, tudo isso com conectividade sem fio para a Internet em âmbito mundial e alta largura de banda. Esse dispositivo e a maneira de conectá-lo são os temas da terceira geração de telefonia móvel. Para obter mais informações, consulte (Huber *et al.*, 2000; e Sarikaya, 2000).

Em 1992, a ITU tentou ser um pouco mais específica em relação a esse sonho e apresentou um projeto para alcançá-lo, denominado **IMT-2000**, onde IMT significava **International Mobile Telecommunications** (telecomunicações móveis internacionais). O número 2000 tinha três significados: (1) o ano em que o sistema deveria entrar em serviço, (2) a freqüência na qual ele deveria operar (em MHz), e (3) a largura de banda que o serviço deveria ter (em kHz).

Esse valor não foi alcançado em nenhum dos três aspectos. Em primeiro lugar, não havia nada implementado em 2000. A ITU recomendou que todos os governos reservassem uma parte do espectro a 2 GHz, de forma que os dispositivos pudessem passar sem problemas de um país para outro. A China reservou a largura de banda exigida, mas nenhum outro país o fez. Por fim, reconheceu-se que uma largura de banda de 2 Mbps não era viável naquele momento para usuários com uma mobilidade *muito* grande (em virtude da dificuldade para emitir handoffs com rapidez suficiente). Uma escolha mais realista é 2 Mbps para usuários estacionários em recintos fechados (que competirão frontalmente com o ADSL), 384 kbps para pessoas andando a pé e 144 kbps para conexões em automóveis. Não obstante, a área inteira da **3G**, como ela é chamado, é um enorme caldeirão em intensa atividade. A terceira geração pode ser um pouco menor do que se esperava originalmente e talvez

chegue um pouco atrasada, mas sem dúvida acontecerá.

Os serviços básicos que a rede IMT-2000 deverá oferecer a seus usuários são:

1. Transmissão de voz de alta qualidade.
2. Serviço de mensagens (substituindo correio eletrônico, fax, SMS, bate-papo etc.).
3. Multimídia (reprodução de música, exibição de vídeos, filmes, televisão etc.).
4. Acesso à Internet (navegação na Web, incluindo páginas com áudio e vídeo).

Outros serviços poderiam ser: videoconferência, telepresença, jogos em grupos e m-commerce (comércio móvel, bastando utilizar seu telefone no caixa para pagar as compras feitas em uma loja). Além disso, todos esses serviços devem estar disponíveis em âmbito mundial (com conexão automática via satélite, quando não for possível localizar nenhuma rede terrestre), de forma instantânea (sempre ativos) e com garantias de qualidade de serviço.

A ITU previu uma única tecnologia mundial para o IMT-2000, de forma que os fabricantes fossem capazes de construir um único dispositivo que pudesse ser vendido e utilizado em qualquer lugar do mundo (como os reprodutores de CDs e computadores, mas diferente dos telefones celulares e dos televisores). Ter uma única tecnologia também facilitaria bastante a vida para os operadores de redes e encorajaria mais pessoas a usarem os serviços. Guerras de formatos, como a que ocorreu entre o Betamax e o VHS quando surgiram os primeiros gravadores de vídeo, não constituem uma boa idéia.

Várias propostas foram feitas e, após uma seleção, elas se reduziram a duas. A primeira, o **W-CDMA (Wideband CDMA — CDMA de banda larga)**, foi proposto pela Ericsson. Esse sistema utiliza um espectro de dispersão de seqüência direta do que tipo que descrevemos. Ele funciona em uma largura de banda de 5 MHz e foi projetado para interoperar com redes GSM, embora não apresente compatibilidade retroativa em relação ao GSM. Contudo, ele tem a propriedade de

permitir a um chamador deixar uma célula W-CDMA e entrar em uma célula GSM sem perder a chamada. Esse sistema foi adotado pela União Européia, que o chamou **UMTS (Universal Mobile Telecommunications System — sistema universal de telecomunicações móveis).**

O outro concorrente era o **CDMA2000**, proposto pela Qualcomm. Ele também é um projeto de espectro de dispersão de seqüência direta, basicamente uma extensão do IS-95, apresentando compatibilidade retroativa com este último. O CDMA2000 também utiliza uma largura de banda de 5 MHz, mas não foi criado para interoperar com o GSM e não pode efetuar o handoff de chamadas para uma célula GSM (nem para uma célula D-AMPS). Outras diferenças técnicas em relação ao W-CDMA incluem uma taxa de chips diferente, um período de quadro distinto, utilização de um espectro diferente e um modo diferente de realizar a sincronização.

Se os engenheiros da Ericsson e da Qualcomm fossem confinados em uma sala e convidados a apresentar um projeto comum, é bem provável que conseguissem fazê-lo. Afinal de contas, o princípio básico que rege ambos os sistemas é o CDMA em um canal de 5 MHz, e ninguém está disposto a morrer para defender sua taxa de chips preferida. A dificuldade é que o problema real não é de engenharia, mas político (como sempre). A Europa queria um sistema que interoperasse com o GSM; os Estados Unidos queriam um sistema que fosse compatível com um sistema já amplamente desenvolvido nos Estados Unidos (o IS-95). Cada lado também apoiava sua empresa local (a Ericsson está sediada na Suécia; a Qualcomm está na Califórnia). Por fim, a Ericsson e a Qualcomm estavam envolvidas em numerosos processos relacionados a suas respectivas patentes de CDMA.

Em março de 1999, as duas empresas apaziguaram os processos, quando a Ericsson concordou em adquirir a infra-estrutura da Qualcomm. Elas também

chegaram a um acordo quanto a um único padrão de 3G, mas era um padrão com várias opções incompatíveis que, em grande parte, eram apenas documentos relativos às diferenças técnicas. Entretanto, apesar dessas disputas, os dispositivos e serviços 3G provavelmente começarão a aparecer no mercado nos próximos anos.

Muito se tem escrito sobre os sistemas 3G, a maior parte louvando-os como a maior invenção desde o pão fatiado. Algumas referências são (Collins e Smith, 2001; De Vriendt *et al.*, 2002; Harte *et al.*, 2002; Lu, 2002; e Sarikaya, 2000). Porém, algumas vozes discordantes consideram que a indústria está orientada na direção errada (Garber, 2002; e Goodman, 2000).

Enquanto se espera pelo fim da disputa sobre a 3G, algumas operadoras estão dando um passo cauteloso em direção à 3G, chegando ao que se costuma chamar às vezes **2,5G**, embora a identificação 2,1G talvez fosse mais precisa. Um sistema desse tipo é o **EDGE (Enhanced Data rates for GSM Evolution — taxas de dados aperfeiçoadas para evolução do GSM)**, que é simplesmente o GSM com mais bits por baud. O problema é que mais bits por baud também significa mais erros por baud, e assim o EDGE tem nove esquemas diferentes para modulação e correção de erros, que se distinguem pela proporção da largura de banda dedicada à correção dos erros introduzidos pela velocidade mais alta.

Outro esquema de 2,5G é o **GPRS (General Packet Radio Service — serviço geral de rádio de pacotes)**, uma rede de superposição de pacotes sobre o D-AMPS ou o GSM. Ele permite que estações móveis transmitam e recebam pacotes IP em uma célulaque executa um sistema de voz. Quando o GPRS está em operação, alguns slots de tempo em algumas freqüências são reservados para tráfego de pacotes. O número e o local dos slots de tempo podem ser dinamicamente gerenciados pela estação base, dependendo da relação entre o tráfego de voz e de dados na célula.

Os slots de tempo disponíveis estão divididos em vários canais lógicos, usados para propósitos diferentes. A estação base determina que canais lógicos serão mapeados sobre que slots de tempo. Um canal lógico se destina à transferência de pacotes da estação base para alguma estação móvel, com cada pacote indicando seu destinatário. Para enviar um pacote IP, uma estação móvel solicita um ou mais slots de tempo, enviando uma solicitação à estação base. Se a solicitação chegar sem danos, a estação base anunciará a freqüência e os slots de tempo alocados à estação móvel para transmissão do pacote. Depois que o pacote chegar à estação base, ele será transferido para a Internet por uma conexão física. Tendo em vista que o GPRS é apenas uma superposição do sistema de voz existente, ele é, na melhor das hipóteses, uma medida paliativa até a chegada da 3G.

Embora as redes 3G ainda não estejam plenamente desenvolvidas, alguns pesquisadores consideram a 3G um trabalho encerrado e, portanto, sem mais interesse. Essas pessoas já estão trabalhando em sistemas 4G (Berezdivin *et al.*, 2002; Guo e Chaskar, 2002; Huang e Zhuang, 2002; Kellerer *et al.*, 2002; e Misra *et al.*, 2002). Algumas características propostas dos sistemas 4G incluem alta largura de banda, onipresença (conectividade em todo lugar), integração uniforme com redes fisicamente conectadas e em especial com o IP, gerenciamento adaptável de recursos e do espectro, rádios por software e alta qualidade de serviço para multimídia.

Por outro lado, estão sendo configurados tantos pontos de acesso a LANs sem fios 802.11 em todo o mundo, que algumas pessoas consideram que a 3G não só não é um trabalho encerrado, mas também que ela está condenada. Segundo essa visão, as pessoas simplesmente passarão de um ponto de acesso 802.11 a outro para permanecerem conectadas. Dizer que a indústria se encontra em um estado de enorme fluxo é uma grande simplificação. Teremos de esperar cerca de 5 anos

para ver o que acontece.

[T2] 2.7 Televisão a cabo

Concluímos o estudo dos sistemas de telefonia fixa e sem fio, com uma quantidade razoável de detalhes. Sem dúvida, ambos desempenharão um papel fundamental nas redes futuras. Porém, há uma alternativa disponível para redes fixas que está se tornando uma opção importante: as redes de televisão a cabo. Muitas pessoas já recebem seu serviço telefônico e da Internet por cabo, e as operadoras de serviços de cabo estão trabalhando ativamente para aumentar sua participação no mercado. Nas próximas seções, vamos examinar com mais detalhes a televisão a cabo como um sistema de redes e vamos compará-lo com os sistemas de telefonia que acabamos de estudar. Para obter mais informações sobre sistemas a cabo, consulte (Laubach *et al.*, 2001; Louis, 2002; Ovadia, 2001; e Smith, 2002).

[T3] 2.7.1 CATV (Community Antenna Television)

A televisão a cabo foi concebida no final dos anos 40 como uma forma de proporcionar melhor recepção às pessoas que vivem em áreas rurais ou montanhosas. No início, o sistema consistia em uma grande antena situada no alto de uma colina para captar o sinal de televisão que se propaga pelo ar, um amplificador chamado **head end** para reforçá-lo e um cabo coaxial para distribuí-lo pelas casas das pessoas, como ilustra a Figura 2.46.

[arte: imagem original da p. 169]

[Dísticos]

[1]Antena para captar sinais distantes

[2]Head end

[3]Cabo de descida

[4]Derivação

[5]Cabo coaxial

[F]Figura 2.46

[FL] Um antigo sistema de televisão a cabo

Nos primeiros anos, a televisão a cabo era chamada **CATV (Community Antenna Television — televisão de antena da comunidade).** Sua operação era muito simples; qualquer pessoa que tivesse alguma prática em eletrônica era capaz de instalar um serviço para sua cidade, e os usuários se reuniam para pagar os custos do serviço. À medida que o número de assinantes crescia, outros cabos eram conectados ao cabo original, sendo acrescentados outros amplificadores conforme a necessidade. A transmissão era unidirecional, do head end para os usuários. Em 1970, havia milhares de sistemas independentes.

Em 1974, a Time, Inc., lançou um novo canal, denominado Home Box Office, com novo conteúdo (filmes) e distribuído somente por cabo. Seguiram-se outros canais dedicados apenas a notícias, esportes, culinária e muitos outros tópicos. Esse desenvolvimento ocasionou duas mudanças na indústria. Primeiro, as grandes corporações começaram a adquirir os sistemas a cabo existentes e a estender novos cabos para conquistar novos assinantes. Em segundo lugar, agora havia necessidade de conectar vários sistemas, freqüentemente de cidades distantes, a fim de distribuir o conteúdo dos novos canais de televisão a cabo. As empresas de televisão a cabo começaram a estender cabos entre suas cidades para conectar todas elas em um único sistema. Esse padrão era análogo ao que aconteceu na indústria de telefonia 80 anos antes, com a conexão de estações finais anteriormente isoladas para tornar possível a comunicação interurbana.

[T3] 2.7.2 Internet por cabo

Com o passar dos anos, o sistema de televisão a cabo cresceu, e os cabos entre as várias cidades foram substituídos por fibra óptica de alta largura de banda, de forma semelhante ao que aconteceu no sistema telefônico. Um sistema com fibra nas linhas principais e cabo coaxial nas ligações para as residências é chamado sistema **HFC (Hybrid Fiber Coax — sistema híbrido de cabo coaxial e fibra).** Os conversores eletro-ópticos que constituem a interface entre as partes óptica e elétrica do sistema são chamados **nós de fibra.** Pelo fato de a largura de banda da fibra ser muito maior que a dos cabos coaxiais, um nó de fibra pode alimentar vários cabos coaxiais. A Figura 2.47(a) mostra uma parte de um moderno sistema HFC.

Nos últimos anos, muitas operadoras de televisão a cabo decidiram entrar no ramo de acesso à Internet e, muitas vezes, no ramo de telefonia. No entanto, diferenças técnicas entre as instalações de cabo e de telefonia têm efeito sobre o que deve ser realizado para se alcançar esses objetivos. Por exemplo, todos os amplificadores unidirecionais do sistema têm de ser substituídos por amplificadores bidirecionais.

No entanto, há outra diferença entre o sistema HFC da Figura 2.47(a) e o sistema telefônico da Figura 2.47(b), e essa diferença é muito mais difícil de remover. Nos bairros, um único cabo é compartilhado por muitas casas, enquanto no sistema telefônico, cada casa tem seu próprio loop local privado. Quando é utilizado para difusão de televisão, esse compartilhando não tem grande importância, pois todos os programas são transmitidos no cabo e não importa se existem 10 ou 10.000 espectadores. Por outro lado, quando o mesmo cabo é usado para acesso à Internet, faz uma grande diferença a existência de 10 ou de 10.000 usuários. Se um usuário decidir baixar um arquivo muito grande, essa largura de banda estará potencialmente sendo retirada de outros usuários. Quanto mais usuários, maior a competição pela largura de banda. O sistema de telefonia não tem essa

propriedade específica: a transferência de um grande arquivo por uma linha ADSL não reduz a largura de banda do seu vizinho. Por outro lado, a largura de banda do cabo coaxial é muito mais alta do que a dos pares trançados.

A estratégia usada pela indústria de serviços a cabo para resolver esse problema é desmembrar cabos longos e conectar cada um deles diretamente a um nó de fibra. A largura de banda do head end até cada nó de fibra é efetivamente infinita, e assim como não existem muitos assinantes em cada segmento de cabo, o volume de tráfego é gerenciável. Os cabos típicos atuais conectam de 500 a 2000 casas; porém, à medida que um número cada vez maior de pessoas adquire um serviço da Internet por cabo, a carga pode se tornar grande demais, exigindo mais divisões e mais nós de fibra.

[arte: imagem original da p. 171]

[Dísticos]

[1] Switch

[2] Tronco de fibra de alta largura de banda

[3] Nó de fibra

[4] Cabo coaxial

[5] Head end

[6] Derivação

[7] Fibra

[8] Casa

[9] (a)

[10] Estação interurbana

[11] Tronco de fibra de alta largura de banda

[12] Estação final

[13] Loop local

[14] Casa

[15] Fibra

[16] Fios de cobre de par trançado

[17] (b)

[F]Figura 2.47

[FL] (a) Televisão a cabo. (b) O sistema de telefonia fixa

[T3] 2.7.3 Alocação do espectro

Eliminar todos os canais de TV e usar a infra-estrutura de cabo estritamente para acesso à Internet provavelmente iria gerar um número razoável de clientes irados; assim, as empresas de televisão a cabo hesitam em fazê-lo. Além disso, a maioria das cidades tem uma regulamentação bastante pesada sobre o que é transmitido pelo cabo, e portanto as operadoras de serviços de cabo não teriam permissão para fazer isso, ainda que desejassem. Como conseqüência, elas precisaram encontrar um modo de fazer a televisão e a Internet coexistirem no mesmo cabo.
Os canais de televisão a cabo da América do Norte ocupam normalmente a região de 54 a 550 MHz (com exceção do rádio FM, que ocupa a faixa de 88 a 108 MHz). Esses canais têm 6 MHz de largura, incluindo as bandas de proteção. Na Europa, a extremidade inferior em geral é de 65 MHz, e os canais têm de 6 a 8 MHz de largura, devido à maior resolução exigida pelos sistemas PAL e SECAM mas, nos outros aspectos, o esquema de alocação é semelhante. A parte baixa da banda não é usada. Os cabos modernos também operam bem acima de 550 MHz, chegando com freqüência a 750 MHz ou mais. A solução escolhida foi introduzir upstream na banda de 5 a 42 MHz (um pouco mais alta na Europa) e usar as freqüências na extremidade alto para o fluxo downstream. O espectro dos serviços de cabo é ilustrado na Figura 2.48.

[arte: imagem original da p. 172]

[Dísticos]

[1] 5 42 54 88

0 108 550 750 MHz

[2] Dados upstream TV FM TV Dados downstream

[3] Freqüências upstream Freqüências downstream

[F]Figura 2.48

[FL] Alocação de freqüências em um sistema típico de TV a cabo usado para acesso à Internet

Observe que, como os sinais de televisão são todos downstream, é possível usar amplificadores upstream que só funcionam na região de 5 a 42 MHz e amplificadores downstream que só funcionam na freqüência de 54 MHz e acima desta, como mostra a figura. Desse modo, obtemos uma assimetria nas larguras de banda upstream e downstream, porque está disponível uma parte maior do espectro acima da faixa de TV do que abaixo dela. Por outro lado, a maior parte do tráfego provavelmente será downstream, e assim as operadoras de serviços a cabo não ficarão insatisfeitas com essa fatalidade. Como vimos antes, em geral as companhias telefônicas oferecem um serviço DSL assimétrico, embora não tenham nenhuma razão técnica para fazê-lo.

Cabos coaxiais longos não são melhores para transmissão de sinais digitais que loops locais longos; portanto, a modulação analógica também é necessária aqui. O esquema habitual é tomar cada canal downstream de 6 MHz ou 8 MHz e modulá-lo com QAM-64 ou, se a qualidade do cabo for excepcionalmente boa, com QAM-256. Com um canal de 6 MHz e QAM-64, obtemos cerca de 36 Mbps. Quando o overhead é subtraído, a carga útil líquida é de aproximadamente 27 Mbps. Com QAM-256, a carga útil líquida é de cerca de 39 Mbps. Os valores europeus são 1/3 maiores.

No caso do fluxo upstream, nem mesmo a QAM-64 funciona bem. Existe ruído

demais de microondas terrestres, rádios da faixa do cidadão e outras fontes; assim, é usado um esquema mais conservador — o QPSK. Esse método (mostrado na Figura 2.25) produz 2 bits por baud em lugar dos 6 ou 8 bits que a QAM oferece nos canais downstream. Conseqüentemente, a assimetria entre a largura de banda upstream e a largura de banda downstream é muito maior do sugere a Figura 2.48.

Além de atualizar os amplificadores, a operadora também tem de atualizar o head end, que deve passar de um amplificador não inteligente para um sistema inteligente de computador digital com uma interface de fibra de alta largura de banda para um ISP. Com freqüência, o nome também é atualizado, de "head end" para **CMTS (Cable Modem Termination System — sistema de terminação de modem a cabo).** No texto a seguir, evitaremos realizar uma atualização de nome e continuaremos usando o termo tradicional "head end".

[T3] 2.7.4 Modems a cabo

O acesso à Internet exige um modem a cabo, um dispositivo que tem duas interfaces: uma para o computador e uma para a rede a cabo. Nos primeiros anos da Internet a cabo, cada operadora tinha um modem a cabo patenteado (proprietário), instalado por um técnico da empresa de serviços a cabo. Entretanto, logo se tornou aparente que um padrão aberto criaria um mercado de modems a cabo competitivo e reduziria os preços, encorajando assim o uso do serviço. Além disso, fazer os clientes comprarem e instalarem eles próprios os modems a cabo (como fazem no caso dos modems telefônicos V.9x) eliminaria os temidos problemas de assistência técnica.

Conseqüentemente, as maiores operadoras de serviço a cabo se uniram a uma empresa chamada CableLabs para produzir um padrão de modem a cabo e testar a compatibilidade dos produtos. Esse padrão, chamado **DOCSIS (Data Over Cable**

Service Interface Specification — especificação de interface de serviços de dados por cabo) está apenas começando a substituir os modems patenteados. A versão européia é chamada **EuroDOCSIS**. Porém, nem todas as operadoras de serviços a cabo apreciam a idéia de um padrão, pois muitas delas estavam ganhando um bom dinheiro arrendando seus modems a seus clientes cativos. Um padrão aberto com dezenas de fabricantes vendendo modems a cabo em lojas encerra essa prática lucrativa.

A interface entre o modem e o computador é simples. Em geral, ela é feita por Ethernet de 10 Mbps (ou, ocasionalmente, por USB). No futuro, o modem inteiro poderia ser uma pequena placa conectada ao computador, da mesma maneira que os modems internos V.9x.

A outra extremidade é mais complicada. Grande parte do padrão lida com engenharia de rádio, um assunto que está muito além do escopo deste livro. A única parte que vale a pena mencionar aqui é que os modems a cabo, como os modems ADSL, estão sempre ativos. Eles estabelecem uma conexão ao serem ligados e mantêm essa conexão durante o tempo em que permanecem ligados, porque as operadoras de serviços a cabo não cobram tarifas pelo tempo de conexão.

Para entender melhor como esses modems funcionam, vamos ver o que acontece quando um modem a cabo é conectado e ligado. O modem percorre os canais downstream procurando por um pacote especial emitido periodicamente pelo head end para fornecer parâmetros do sistema aos modems que acabaram de se conectar. Ao encontrar esse pacote, o novo modem anuncia sua presença em um dos canais upstream. O head end responde atribuindo o modem a seus canais upstream e downstream. Essas atribuições podem ser alteradas mais tarde, se o head end julgar necessário equilibrar a carga.

Em seguida, o modem determina sua distância até o head end, enviando-lhe um

pacote especial e verificando quanto tempo demora para receber a resposta. Esse processo é chamado **verificação do alcance.** É importante para o modem conhecer sua distância, a fim de se acomodar ao modo de operação dos canais upstream e obter a sincronização correta. Eles se dividem no tempo em **minislots.** Cada pacote upstream deve caber em um ou mais minislots consecutivos. O head end anuncia periodicamente o início de uma nova rodada de minislots, mas o tiro de partida não é ouvido em todos os modems ao mesmo tempo, devido ao tempo de propagação no cabo. Conhecendo a que distância está do head end, cada modem pode calcular há quanto tempo realmente começou o primeiro minislot. A extensão do minislot depende da rede. Uma carga útil típica é de 8 bytes.

Durante a inicialização, o head end também atribui cada modem a um minislot, que será usado para solicitar largura de banda upstream. Como regra, vários modems serão atribuídos ao mesmo minislot, o que resulta em disputa. Quando um computador quer enviar um pacote, ele transfere o pacote ao modem, que então solicita o número necessário de minislots. Se a solicitação for aceita, o head end colocará uma confirmação no canal downstream, informando ao modem quais minislots foram reservados para seu pacote. O pacote é então enviado, a partir do minislot alocado a ele. Pacotes adicionais podem ser solicitados com a utilização de um campo no cabeçalho.

Por outro lado, se houver disputa pelo minislot solicitado, não haverá nenhuma confirmação, e o modem simplesmente irá esperar um tempo aleatório e tentar de novo. Após cada falha sucessiva, o tempo de randomização (aleatoriedade) é duplicado. (Para os leitores que já estão um pouco familiarizados com as redes, esse algoritmo é simplesmente é o slotted ALOHA com o recuo binário exponencial. A Ethernet não pode ser usada em redes a cabo, porque as estações não conseguem detectar o meio. Voltaremos para essas questões no Capítulo 4.)

Os canais downstream são gerenciados de modo diferente dos canais upstream.

Por um lado, só existe um transmissor (o head end) e assim não há disputa nem a necessidade de minislots que, na realidade, são apenas a multiplexação estatística por divisão de tempo. Por outro lado, o tráfego downstream em geral é muito maior que o tráfego upstream, e então é usado um tamanho de pacote fixo, igual a 204 bytes. Uma parte desse código é um código de correção de erros de Reed-Solomon e algumas outras fontes de overhead, restando uma carga útil do usuário igual a 184 bytes. Esses números foram escolhidos para manter a compatibilidade com a televisão digital usando MPEG-2, de forma que os canais de TV e os canais de dados downstream sejam formatados de maneira idêntica. As conexões lógicas estão representadas na Figura 2.49.

Voltando à inicialização do modem, depois que o modem conclui a verificação do alcance e recebe seu canal upstream, o canal downstream e as atribuições de minislots, ele está livre para começar a transmitir pacotes. O primeiro pacote enviado se destina ao ISP e solicita um endereço IP, que é atribuído dinamicamente com o uso de um protocolo chamado DHCP, que estudaremos no Capítulo 5. Ele também solicita e obtém do head end a hora exata do dia.

[arte: imagem original da p. 175]

[Dísticos]

[1] ISP Fibra Head end

[2] Cabo coaxial

[3] Canal downstream sem disputa: 27 Mbps usando QAM-64 e carga útil de 184 bytes

[4] Modem

[5] Canal upstream com disputa: 9 Mbps usando QPSK e minislots de 8 bytes

[6] Pacote

[F]Figura 2.49

[FL] Detalhes típicos dos canais upstream e downstream na América do Norte

A próxima etapa envolve a segurança. Como o cabo é um meio compartilhado, qualquer pessoa que queira ter o trabalho de fazê-lo pode ler todo o tráfego que está sendo transmitido. Para evitar que alguém espione seus vizinhos (literalmente), todo tráfego é criptografado em ambos os sentidos. Uma parte do procedimento de inicialização envolve o estabelecimento de chaves de criptografia. À primeira vista, poderíamos imaginar que fazer dois estranhos, o head end e o modem, estabelecerem uma chave secreta em plena luz do dia com milhares de pessoas observando seria uma tarefa impossível. Na verdade, não é impossível, mas teremos de esperar até o Capítulo 8 para explicar como fazê-lo (resumindo, devemos usar o algoritmo de Diffie-Hellman).

Finalmente, o modem tem de se conectar e fornecer seu identificador exclusivo pelo canal seguro. Nesse momento, a inicialização se completa. Agora, o usuário pode se conectar ao ISP e começar a trabalhar.

Há muito mais a ser dito sobre modems a cabo. Algumas referências relevantes são (Adams e Dulchinos, 2001; Donaldson e Jones, 2001; e Dutta-Roy, 2001).

[T3] 2.7.5 ADSL *versus* cabo

O que é melhor, ADSL ou cabo? Isso é como perguntar qual sistema operacional, qual linguagem ou qual religião é melhor. A resposta depende da pessoa a quem você faz a pergunta. Vamos comparar alguns aspectos das tecnologias ADSL e de cabo. Ambas utilizam fibra no backbone, mas diferem nas extremidades. O sistema a cabo utiliza cabo coaxial, enquanto o ADSL usa par trançado. A capacidade teórica de transporte do cabo coaxial é centenas de vezes maior que a do par trançado. Contudo, a capacidade total do cabo não está disponível para usuários de dados, porque grande parte da largura de banda do cabo é desperdiçada com material inútil, como programas de televisão.

Na prática, é difícil generalizar a respeito da capacidade efetiva. Os provedores de ADSL fazem declarações específicas sobre a largura de banda (por exemplo, 1 Mbps downstream, 256 kbps upstream) e, em geral, alcançam cerca de 80% desses valores de forma consistente. Os provedores de serviços a cabo não fazem nenhuma afirmação, porque a capacidade efetiva depende da quantidade de pessoas atualmente ativas no segmento de cabo do usuário. Algumas vezes, talvez ela seja melhor que a ADSL, e outras vezes, pode ser pior. Entretanto, o que talvez incomode é a imprevisibilidade. O fato de ter um ótimo serviço em um minuto não garante um ótimo serviço no minuto seguinte, pois o maior devorador de largura de banda da cidade pode ter acabado de ligar seu computador nesse momento.

À medida que um sistema ADSL conquista mais usuários, seus números crescentes têm pouco efeito sobre os usuários existentes, pois cada usuário tem uma conexão dedicada. No caso dos sistemas a cabo, quando mais assinantes se inscrevem para receber o serviço da Internet, o desempenho diminui para os usuários atuais. O único remédio é a operadora dos serviços a cabo dividir os cabos ocupados e conectar cada um deles diretamente a um nó de fibra. Isso custa tempo e dinheiro, e portanto há pressões comerciais para evitar fazê-lo.

A propósito, já estudamos outro sistema com um canal compartilhado semelhante ao cabo: o sistema de telefonia móvel. Também nesse caso, um grupo de usuários, que poderíamos chamar companheiros de célula, compartilham um volume fixo de largura de banda. Normalmente, ele é dividido de modo rígido em blocos fixos entre os usuários ativos pela FDM e pela TDM, porque o tráfego de voz é bastante suave. Porém, no caso do tráfego de dados, essa divisão rígida é muito ineficiente porque, com freqüência, os usuários de dados estão ociosos e, nesse caso, a largura de banda reservada para eles é desperdiçada. Apesar disso, nesse aspecto o acesso a sistemas de cabo é mais

parecido com o sistema de telefonia móvel do que com o sistema fixo.

A disponibilidade é um ponto no qual as tecnologias ADSL e de cabo diferem. Todo mundo tem um telefone, mas nem todos os usuários estão próximos o bastante de sua estação final para receber o serviço ADSL. Por outro lado, nem todos têm a tecnologia de cabo mas, se você tem essa tecnologia e a empresa fornece acesso à Internet, é possível obtê-lo, pois a distância até o nó de fibra ou até o head end não é problema. Também vale a pena notar que, como a tecnologia de cabo teve início como um meio de distribuição de televisão, poucas empresas têm esse recurso.

Sendo um meio ponto a ponto, a ADSL é inerentemente mais segura que o cabo. Qualquer usuário de serviços de cabo pode ler com facilidade todos os pacotes que passam pelo cabo. Por essa razão, qualquer provedor de serviços a cabo decente irá criptografar todo o tráfego em ambos os sentidos. Apesar disso, é mais seguro impedir que seu vizinho receba as mensagens criptografadas destinadas a você do que impedir que ele receba absolutamente qualquer mensagem.

Em geral, o sistema de telefonia é mais confiável do que o sistema a cabo. Por exemplo, ele tem potência de reserva e continua a funcionar normalmente mesmo durante uma queda de energia. No caso dos sistemas a cabo, se houver uma falha de energia em qualquer amplificador ao longo da cadeia, todos os usuários situados abaixo dele serão instantaneamente desconectados.

Por fim, a maioria dos provedores de ADSL oferece a opção de escolher ISPs. Às vezes, eles são até mesmo obrigados a fazer isso por lei, o que nem sempre acontece no caso das operadoras de serviços a cabo.

A conclusão é que a ADSL e o cabo são muito mais semelhantes do que diferentes. Eles oferecem um serviço comparável e, à medida que a competição entre as duas tecnologias se aquecer, provavelmente também oferecerão preços

comparáveis.

[T2] 2.8 Resumo

A camada física é a base de todas as redes. A natureza impõe dois limites fundamentais sobre todos os canais, que determinam sua largura de banda. Esses limites são o limite de Nyquist, que lida com canais sem ruídos, e o limite de Shannon, que trata de canais com ruídos.

Os meios de transmissão podem ser guiados ou não guiados. Os principais meios guiados são o par trançado, o cabo coaxial e a fibra óptica. Dentre os meios não guiados estão o rádio, as microondas, os raios infravermelhos e os raios laser. Um sistema de transmissão com boas perspectivas de êxito é a comunicação por satélite, em especial os sistemas LEO.

Um elemento fundamental em muitas redes geograficamente distribuídas é o sistema telefônico. Seus principais componentes são os loops locais, troncos e switches. Os loops locais são circuitos analógicos de par trançado que exigem modems para a transmissão de dados digitais. A ADSL oferece velocidades de até 50 Mbps, dividindo o loop local em muitos canais virtuais e modulando cada um deles separadamente. Os loops locais sem fios são outro desenvolvimento que devemos observar, em especial o LMDS.

Os troncos são digitais e podem ser multiplexados de várias formas, incluindo FDM, TDM e WDM. A comutação de circuitos e a comutação de pacotes são importantes.

Para aplicações móveis, o sistema telefônico fixo não é adequado. Os telefones celulares estão sendo amplamente utilizados hoje para voz e em breve serão intensamente empregados na transmissão de dados. A primeira geração era analógica, dominada pelo AMPS. A segunda geração era digital, sendo D-AMPS, GSM e CDMA as principais opções. A terceira geração será digital e se baseará no

CDMA de banda larga.

Um sistema alternativo para acesso de rede é o sistema de televisão a cabo, que evoluiu gradualmente desde uma antena comunitária até chegar ao sistema híbrido de cabo coaxial e fibra. Potencialmente, ele oferece largura de banda muito alta, mas a largura de banda real disponível na prática depende muito do número de outros usuários ativos no momento e do que eles estão fazendo.

[T2] Problemas

1. Calcule os coeficientes de Fourier para a função $f(t) = t$ $(0 \leq t \leq 1)$.

2. Um canal sem ruído de 4 kHz tem uma amostra a cada 1 ms. Qual é a taxa máxima de dados desse canal?

3. Os canais de televisão têm 6 MHz. Quantos bits/s poderão ser enviados, se forem usados sinais digitais de quatro níveis? Suponha um canal sem ruído.

4. Se um sinal binário for enviado sobre um canal de 3 kHz cuja relação sinal/ruído é de 20 dB, qual será a taxa máxima de dados que poderá ser alcançada?

5. Qual é a relação sinal/ruído necessária para colocar uma portadora T1 em uma linha de 50 kHz?

6. Qual é a diferença entre uma estrela passiva e um repetidor ativo em uma rede de fibra óptica?

7. Qual é a largura de banda existente em 0,1 mícron de espectro em um comprimento de onda de 1 mícron?

8. Queremos enviar uma seqüência de imagens de tela de computador por fibra óptica. A tela tem 480×640 pixels, e cada pixel tem 24 bits. Há 60 imagens de tela por segundo. Qual é a largura de banda necessária, e quantos micra de comprimento de onda são necessários para essa banda a 1,30 micra?

9. O teorema de Nyquist também se aplica à fibra óptica, ou somente ao fio de

cobre?

10. Na Figura 2.6, a banda da esquerda é mais estreita do que as outras. Por quê?

11. Em geral, as antenas de rádio funcionam melhor quando o diâmetro da antena é igual ao comprimento de onda das ondas de rádio. Uma variação razoável para o diâmetro das antenas é de 1 cm a 5 m. Que faixa de freqüências é coberta por esse intervalo?

12. O esmaecimento de vários caminhos é maximizado quando os dois feixes chegam ao destino defasados 180 graus. Que diferença de percurso é necessária para maximizar o esmaecimento (fading) em um enlace de microondas de 1 GHz com 50 Km de extensão?

13. Um feixe de raios laser de 1 mm está orientado para um detector localizado a 100 m de distância do telhado de um edifício. Que desvio angular (em graus) o laser precisa ter antes de perder o detector?

14. Os 66 satélites de baixa órbita do projeto Iridium estão divididos em seis colares em torno da Terra. Na altitude em que eles se encontram, o período é de 90 minutos. Qual é o intervalo médio entre handoffs no caso de um transmissor estacionário?

15. Considere um satélite na altitude dos satélites geoestacionários, mas cujo plano orbital está inclinado em relação ao plano equatorial por um ângulo [ver símbolo]. Para um usuário estacionário na superfície da Terra na latitude [ver símbolo] norte, esse satélite parecerá imóvel no céu? Se não, descreva seu movimento.

16. Quantos códigos de estações finais existiam antes de 1984, quando cada estação final era identificada por seu código de área de três dígitos e pelos três primeiros dígitos do número local? Os códigos de área se iniciavam com um dígito no intervalo de 2 a 9, tinham 0 ou 1 como o segundo e terminavam com qualquer dígito. Os dois primeiros dígitos de um número local sempre estavam

no intervalo de 2 a 9. O terceiro dígito podia ser qualquer dígito.

17. Usando *somente* os dados fornecidos no texto, qual é o número máximo de telefones que o sistema existente nos Estados Unidos pode admitir sem alterar o plano de numeração ou acrescentar equipamentos adicionais? Essa quantidade de telefones poderia de fato ser alcançada? Para fins deste problema, um computador ou um equipamento de fax é considerado um telefone. Suponha que só exista um dispositivo por linha de assinante.

18. Um sistema telefônico simples consiste em duas estações finais e uma única estação interurbana, à qual cada estação final está conectada por um tronco full-duplex de 1 MHz. Um telefone comum é usado para fazer quatro ligações em um dia útil de 8 horas. A duração média de cada chamada é de 6 minutos. 10% das chamadas são interurbanas (ou seja, passam pela estação interurbana). Qual é o número máximo de telefones que uma estação final pode aceitar? (Suponha 4 kHz por circuito.)

19. Uma companhia telefônica regional tem 10 milhões de assinantes. Cada um de seus telefones está conectado a uma estação central por um fio de cobre de par trançado. O comprimento médio desses pares trançados é 10 km. Quanto vale o cobre contido nos loops locais? Suponha que a seção transversal de cada fio seja um círculo com 1 mm de diâmetro, que a densidade específica do cobre seja 9,0 gramas/cm^3 e que o cobre seja vendido ao preço de 3 dólares por quilograma.

20. Um oleoduto é um sistema simplex, um sistema half-duplex, um sistema full-duplex ou nenhum dos anteriores?

21. O custo de um microprocessador rápido diminuiu tanto que agora é possível incluir um em cada modem. De que maneira isso afeta o tratamento de erros na linha telefônica?

22. Um diagrama de constelação de modems semelhante ao da Figura 2.25 tem

pontos de dados nas seguintes coordenadas: (1,1), (1,-1), (-1,1) e (-1,-1). Quantos bps um modem com esses parâmetros pode alcançar a uma taxa de transmissão de 1.200 bauds?

23. Um diagrama de constelação de modem semelhante ao da Figura 2.25 tem pontos de dados em (0, 1) e (0, 2). O modem utiliza modulação de fase ou modulação de amplitude?

24. Em um diagrama de constelação, todos os pontos estão em um círculo com centro na origem. Que espécie de modulação está sendo usada?

25. Quantas freqüências um modem QAM-64 full-duplex utiliza?

26. Um sistema ADSL que usa DMT aloca 3/4 dos canais de dados disponíveis para o enlace downstream. Ele utiliza modulação QAM-64 em cada canal. Qual é a capacidade do enlace downstream?

27. No exemplo de LMDS de quatro setores da Figura 2.30, cada setor tem seu próprio canal de 36 Mbps. De acordo com a teoria do enfileiramento, se o canal estiver 50% carregado, o tempo de enfileiramento será igual ao tempo de transferência. Sob essas condições, quanto tempo ele levará para baixar uma página da Web de 5 KB? Quanto tempo ele levará para baixar a página sobre uma linha ADSL de 1 Mbps? E sobre um modem de 56 kbps?

28. Dez sinais, cada um exigindo 4000 Hz, são multiplexados em um único canal utilizando FDM. Qual é a largura de banda mínima exigida para o canal multiplexado? Suponha que as bandas de proteção tenham 400 Hz de largura.

29. Por que o tempo de amostragem do PCM foi definido como 125 µs?

30. Qual é o percentual de overhead em uma portadora T1; ou seja, que percentagem dos 1,544 Mbps não é entregue ao usuário final?

31. Compare a taxa máxima de dados de um canal sem ruído de 4 kHz usando:

(a) Codificação analógica (por exemplo, QPSK) com 2 bits por amostra.

(b) O sistema PCM T1.

32. Se um sistema de portadora T1 apresentar uma falha e perder o controle de onde está, ele tentará se sincronizar novamente usando o primeiro bit de cada quadro. Quantos quadros terão de ser examinados em média para que seja feita a nova sincronização com uma probabilidade de erro de 0,001?

33. Qual é a diferença, se houver, entre a parte de demodulador de um modem e a parte de codificador de um codec? (Afinal, ambos convertem sinais analógicos em sinais digitais.)

34. Um sinal é transmitido digitalmente sobre um canal sem ruído de 4 kHz com uma amostra a cada 125 µs. Quantos bits por segundo são realmente enviados por cada um destes métodos de codificação?

(a) Padrão CCIIT de 2,048 Mbps.

(b) DPCM com um valor de sinal relativo de 4 bits.

(c) Modulação delta.

35. Uma onda senoidal pura de amplitude A é codificada com o uso da modulação delta, com x amostras/s. Uma saída igual a +1 corresponde a uma mudança de sinal de $+A/8$, e um sinal de saída de –1 corresponde a uma mudança de sinal de $-A/8$. Qual é a freqüência mais alta que pode ser rastreada sem erro cumulativo?

36. Os clocks da SONET têm uma taxa de tração de aproximadamente uma parte em 10^9. Quanto tempo a tração leva para igualar a largura de 1 bit? Quais são as implicações desse cálculo?

37. Na Figura 2.37, a taxa de dados do usuário para OC-3 é de 148,608 Mbps. Mostre como esse número pode ser derivado dos parâmetros OC-3 da SONET.

38. Para acomodar taxas de dados mais baixos que STS-1, a SONET tem um sistema de tributários virtuais (VT). Um VT é uma carga útil parcial que pode ser inserida em um quadro STS-1 e combinada com outras cargas úteis parciais para preencher o quadro de dados. O VT1.5 utiliza três colunas, o VT2 usa 4 colunas,

o VT3 usa 6 colunas e o VT6 usa 12 colunas de um quadro STS-1. Qual VT pode acomodar:

(a) Um serviço DS-1 (1,544 Mbps)?

(b) Um serviço europeu CEPT-1 (2,048 Mbps)?

(c) Um serviço DS-2 (6,312 Mbps)?

39. Qual é a diferença essencial entre a comutação de mensagens e a comutação de pacotes?

40. Qual é a largura de banda disponível para o usuário em uma conexão OC-12c?

41. Três redes de comutação de pacotes possuem *n* nós cada uma. A primeira rede tem uma topologia em estrela com um switch central, a segunda é um anel (bidirecional) e a terceira é totalmente interconectada, com um fio interligando cada nó. Quais são as opções de caminhos de transmissão em hops no melhor caso, no caso médio e no pior caso?

42. Compare o retardo no envio de uma mensagem de *x* bits sobre um caminho de *k* hops em uma rede comutada por circuitos e em uma rede comutada por pacotes (levemente carregada). O tempo de configuração de circuitos é *s* segundos, o retardo de propagação é *d* segundos por hop, o tamanho do pacote é *p* bits e a taxa de dados é *b* bps. Sob quais condições a rede de pacotes tem um retardo mais baixo?

43. Suponha que *x* bits dados do usuário tenham de ser transmitidos por um caminho de *k* hops em uma rede comutada por pacotes como uma série de pacotes, cada um contendo *p* bits de dados e *h* bits de cabeçalho, sendo [ver símbolo]. A taxa de bits das linhas é *b* bps e o retardo de propagação é desprezível. Que valor de *p* minimiza o retardo total?

44. Em um sistema telefônico típico com células hexagonais, é proibido reutilizar uma banda de freqüências em uma célula adjacente. Se estão disponíveis 840

freqüências, quantas podem ser utilizadas em uma determinada célula?

45. O layout real de células raramente é tão regular quanto o da Figura 2.41. Mesmo as formas de células individuais em geral são irregulares. Apresente uma razão possível para isso.

46. Faça uma estimativa do número de microcélulas PCS com 100 m de diâmetro que seriam necessárias para cobrir a cidade de San Francisco (120 quilômetros quadrados).

47. Às vezes, quando um usuário móvel cruza o limite de uma célula para outra, a chamada atual é encerrada de forma abrupta, embora todos os transmissores e receptores estejam funcionando perfeitamente. Por quê?

48. O D-AMPS tem uma qualidade de voz muito pior que o GSM. Isso se deve ao fato do D-AMPS ter de apresentar compatibilidade retroativa com o AMPS, enquanto o GSM não tem tal restrição? Se não, qual é a causa do problema?

49. Calcule o número de máximo de usuários que o D-AMPS pode admitir ao mesmo tempo em uma única célula. Faça o mesmo cálculo para o GSM. Explique a diferença.

50. Suponha que *A*, *B* e *C* estejam transmitindo simultaneamente bits 0, usando um sistema CDMA com as seqüências de chips da Figura 2.45(b). Qual é a seqüência de chips resultante?

51. Na discussão sobre ortogonalidade das seqüências de chips do CDMA, afirmamos que, se [ver símbolo] = 0, então [ver símbolo] também é 0. Prove essa afirmação.

52. Considere um modo diferente de observar a propriedade de ortogonalidade das seqüências de chips do CDMA. Cada bit em um par de seqüências pode coincidir ou não. Expresse a propriedade de ortogonalidade em termos de correspondências e não correspondências.

53. Um receptor CDMA recebe os seguintes chips: (–1 +1 –3 +1 –1 –3 +1+1).

Supondo as seqüências de chips definidas na Figura 2.45(b), que estações transmitiram, e quais bits cada uma enviou?

54. Na extremidade baixa, o sistema telefônico tem a forma de estrela, com todos os loops locais em uma vizinhança convergindo em uma estação final. Em contraste, a televisão a cabo consiste em um único cabo longo que passa por todas as casas no mesmo bairro. Suponha que um cabo de TV do futuro fosse uma fibra de 10 Gbps, em vez de fio de cobre. Ele poderia ser usado para simular o modelo de telefonia em que todos têm sua própria linha privada até a estação final? Nesse caso, quantas casas com um telefone poderiam ser conectadas a uma única fibra?

55. Um sistema de TV a cabo tem 100 canais comerciais, todos eles alternando programas com anúncios. Esse sistema é mais parecido com TDM ou FDM?

56. Uma empresa de serviços a cabo decide oferecer acesso à Internet por cabo em um bairro que tem 5000 casas. A empresa utiliza um cabo coaxial e uma alocação de espectro que permite alcançar a largura de banda de 100 Mbps downstream por cabo. Para atrair clientes, a empresa decide garantir pelo menos 2 Mbps de largura de banda downstream para cada casa em qualquer instante. Descreva o que a empresa de serviços a cabo precisa fazer para fornecer essa garantia.

57. Usando a alocação espectral mostrada na Figura 2.48 e as informações dadas no texto, quantos Mbps um sistema de cabo aloca para o tráfego upstream e quantos para o tráfego downstream?

58. Com que velocidade um usuário de serviços de cabo recebe dados, se a rede está ociosa, exceto pela atividade desse usuário?

59. A multiplexação de vários fluxos de dados STS-1, chamados tributários, desempenha um papel importante na SONET. Um multiplexador 3:1 efetua a multiplexação de três tributários STS-1 de entrada em um único fluxo STS-3 de

saída. Essa multiplexação é feita byte a byte, isto é, os três primeiros bytes de saída são os primeiros bytes dos tributários 1, 2 e 3, respectivamente. Os três bytes de saída seguintes são os próximos bytes dos tributários 1, 2 e 3, respectivamente e assim por diante. Escreva um programa que simule esse multiplexador 3:1. Seu programa deve consistir em cinco processos. O processo principal cria quatro processos, um processo para cada um dos três tributários STS-1 e um processo para o multiplexador. Cada processo tributário lê um quadro STS-1 de um arquivo de entrada como uma seqüência de 810 bytes. Eles enviam seus quadros (byte por byte) ao processo multiplexador. O processo multiplexador recebe esses bytes e transmite como saída um quadro STS-3 (byte por byte), gravando-o na saída padrão. Utilize pipes para efetuar a comunicação entre os processos.

[TA1]Capítulo

[TA2]3

[T1] A camada de enlace de dados

Neste capítulo, estudaremos os princípios de projeto da segunda camada, a camada de enlace de dados. Nesse estudo, trataremos de algoritmos que permitem uma comunicação eficiente e confiável entre dois computadores adjacentes no nível da camada de enlace de dados. Por adjacentes, queremos dizer que as duas máquinas estão fisicamente conectadas por meio de um canal de comunicação que funciona conceitualmente como um fio (por exemplo, um cabo coaxial, uma linha telefônica ou um canal sem fio ponto a ponto). A característica de um canal que o torna semelhante a um fio é fato de que os bits são entregues na ordem exata em que são enviados.

Em princípio, você poderá pensar que esse problema é tão trivial que não é necessário um software para tratá-lo, pois a máquina *A* simplesmente coloca os bits no fio e a máquina *B* os retira de lá. Infelizmente, os circuitos de comunicação produzem erros ocasionais. Além disso, eles têm uma taxa de dados finita, e há um retardo de propagação diferente de zero entre o momento em que o bit é enviado e o momento em que ele é recebido. Essas limitações têm implicações importantes para a eficiência da transferência de dados. Os protocolos usados para comunicações devem levar todos esses fatores em consideração. Tais protocolos são o assunto deste capítulo.

Após uma introdução às principais questões de projeto presentes na camada de enlace de dados, começaremos nosso estudo dos protocolos dessa camada verificando a natureza dos erros, suas causas e como eles podem ser detectados e corrigidos. Em seguida, estudaremos uma série de protocolos de complexidade crescente e mostraremos como cada um deles resolve um número cada vez maior

de problemas da camada de enlace de dados. Por fim, concluiremos o capítulo com um exame da modelagem e da correção de protocolos e apresentaremos alguns exemplos de protocolos de enlace de dados.

[T2] 3.1 Questões de projeto da camada de enlace de dados

A camada de enlace de dados executa diversas funções específicas. Dentre elas estão as seguintes:

1. Fornecer uma interface de serviço bem definida à camada de rede.
2. Lidar com erros de transmissão.
3. Regular o fluxo de dados, de tal forma que receptores lentos não sejam atropelados por transmissores rápidos.

Para alcançar esses objetivos, a camada de enlace de dados recebe os pacotes da camada de rede e os encapsula em **quadros** para transmissão. Cada quadro contém um cabeçalho (header) de quadro, um campo de carga útil, que conterá o pacote, e um final (trailer) de quadro, como mostra a Figura 3.1. O gerenciamento de quadros constitui o núcleo das atividades da camada de enlace de dados. Nas próximas seções, examinaremos em detalhes todas as questões mencionadas.

[arte: ver original p. 184]

[Dísticos]

[1]Máquina transmissora

Pacote

[2]Quadro

Cabeçalho Campo de carga útil Final

[3]Máquina receptora

Pacote

[4]Cabeçalho Campo de carga útil Final

[F]Figura 3.1

[FL]Relacionamento entre pacotes e quadros

Embora este capítulo trate explicitamente da camada de enlace de dados e dos protocolos de enlace de dados, muitos dos princípios que estudaremos aqui, como o controle de erros e o controle de fluxo, são encontrados em protocolos de transporte e também em outros protocolos. De fato, em muitas redes, essas funções são encontradas apenas nas camadas superiores e não na camada de enlace de dados. Porém, independente de onde elas são encontradas, os princípios são quase idênticos, e então não importa realmente o lugar em que os estudamos. Na camada de enlace de dados, eles surgem com freqüência em sua forma mais simples e mais pura, o que faz dessa camada um bom lugar para examiná-los em detalhes.

[T3] 3.1.1 Serviços oferecidos à camada de rede

A função da camada de enlace de dados é fornecer serviços à camada de rede. O principal serviço é transferir dados da camada de rede da máquina de origem para a camada de rede da máquina de destino. Na camada de rede da máquina de origem, há uma entidade chamada processo que entrega alguns bits à camada de enlace de dados para transmissão ao destino. A tarefa da camada de enlace de dados é transmitir os bits à máquina de destino, de forma que eles possam ser entregues à camada de rede dessa máquina, como mostra a Figura 3.2(a). A transmissão propriamente dita segue o trajeto descrito na Figura 3.2(b); no entanto, é mais fácil pensar em termos de dois processos da camada de enlace de dados que se comunicam por intermédio de um protocolo de enlace de dados. Por essa razão, utilizaremos implicitamente o modelo da Figura 3.2 (a) em todo este capítulo.

[arte: ver original p. 185]

[Dísticos]

[1]Host 1 Host 2 Host 1 Host 2

[2]4 4 4 4

[3]3 3 3 3

Caminho de dados virtual

[4]2 2 2 2

[5]1 1 1 1

Caminho de dados real

(a) (b)

[F]Figura 3.2

[FL](a) Comunicação virtual. (b) Comunicação real

A camada de enlace de dados pode ser projetada de modo a oferecer diversos serviços, que podem variar de sistema para sistema. Três possibilidades razoáveis oferecidas com freqüência são:

1. Serviço sem conexão e sem confirmação.

2. Serviço sem conexão com confirmação.

3. Serviço orientado a conexões com confirmação.

Agora vamos considerar cada uma dessas possibilidades.

O serviço sem conexão e sem confirmação consiste em fazer a máquina de origem enviar quadros independentes à máquina de destino, sem que a máquina de destino confirme o recebimento desses quadros. Nenhuma conexão lógica é estabelecida antes ou liberada depois do processo. Se um quadro for perdido devido a ruídos na linha, não haverá nenhuma tentativa de detectar a perda ou de recuperá-lo na camada de enlace de dados. Essa classe de serviço é apropriada quando a taxa de erros é muito baixa, e a recuperação fica a cargo de camadas mais altas. Ela também é apropriada para o tráfego em tempo real, no qual, a

exemplo da fala humana, os dados atrasados causam mais problemas que dados recebidos com falhas. A maior parte das LANs utiliza serviços sem conexão e sem confirmação na camada de enlace de dados.

O próximo passo em termos de confiabilidade é o serviço sem conexão com confirmação. Quando esse serviço é oferecido, ainda não há conexões lógicas sendo usadas, mas cada quadro enviado é individualmente confirmado. Dessa forma, o transmissor sabe se um quadro chegou corretamente ou não. Caso não tenha chegado dentro de um intervalo de tempo específico, o quadro poderá ser enviado outra vez. Esse serviço é útil em canais não confiáveis, como os sistemas sem fio.

Talvez valha a pena destacar que oferecer recursos de confirmação no nível da camada de enlace de dados é uma questão de otimização, e não uma exigência. A camada de rede sempre pode enviar um pacote e esperar que ele seja confirmado. Se a confirmação não chegar durante o intervalo do timer, o transmissor poderá enviar a mensagem inteira mais uma vez. O problema dessa estratégia é que, em geral, os quadros têm um comprimento máximo estrito imposto pelo hardware, o que não ocorre com os pacotes da camada de rede. Por exemplo, se o pacote médio for subdividido em, digamos, 10 quadros, e 20% de todos os quadros forem perdidos, o tempo necessário para efetivar a transmissão do pacote com sucesso poderá ser muito longo. Se cada quadro individual for confirmado e retransmitido, os pacotes completos chegarão a seu destino muito mais rapidamente. Em canais confiáveis, como a fibra óptica, o uso de um protocolo de enlace de dados muito sofisticado talvez seja desnecessário mas, em canais sem fio, com sua inerente falta de confiabilidade, o custo compensa.

Voltando aos nossos serviços, o serviço mais sofisticado que a camada de enlace de dados é capaz de oferecer à camada de rede é o serviço orientado a conexões. Com ele, as máquinas de origem e destino estabelecem uma conexão antes de os

dados serem transferidos. Cada quadro enviado pela conexão é numerado, e a camada de enlace de dados garante que cada quadro será de fato recebido. Além disso, essa camada garante que todos os quadros serão recebidos uma única vez e na ordem correta. Por outro lado, com o serviço sem conexão, é concebível que uma confirmação perdida acarrete diversas retransmissões de um quadro e, conseqüentemente, faça com que ele seja recebido várias vezes. Em contraste, os serviços orientados a conexões fornecem aos processos da camada de rede o equivalente a um fluxo de bits confiável.

Quando é usado o serviço orientado a conexões, as transferências passam por três fases distintas. Na primeira fase, a conexão é estabelecida, fazendo-se ambos os lados inicializarem as variáveis e os contadores necessários para controlar os quadros que são recebidos e os que não são. Na segunda fase, um ou mais quadros são realmente transmitidos. Na terceira e última fase, a conexão é desfeita, liberando-se as variáveis, os buffers e os outros recursos usados para mantê-la.

Considere um exemplo típico: uma sub-rede de uma WAN que consiste em roteadores conectados por linhas telefônicas privadas ponto a ponto. Quando um quadro chega a um roteador, o hardware verifica se há erros (utilizando técnicas que estudaremos mais adiante neste capítulo) e depois repassa o quadro ao software da camada de enlace de dados (que pode estar incorporada a um chip na placa de interface de rede). O software da camada de enlace de dados verifica se esse é o quadro esperado e, se for o caso, passa o pacote contido no campo de carga útil (payload) ao software de roteamento. O software de roteamento, por sua vez, seleciona a linha de saída apropriada e repassa o pacote ao software da camada de enlace de dados, que o retransmite. O fluxo existente entre dois roteadores é mostrado na Figura 3.3.

[arte: ver original p. 187]

[Dísticos]

[1]Roteador

[2]Processo da camada de enlace de dados

[3]Processo de roteamento

[4]Quadros aqui

[5]Pacotes aqui

[6]Protocolo de enlace de dados

[7]Linha de transmissão para um roteador

[F]Figura 3.3

[FL]Localização do protocolo de enlace de dados

Com freqüência, o código de roteamento deseja que a tarefa seja executada corretamente, ou seja, que existam conexões confiáveis que preservem a seqüência dos quadros em cada uma das linhas ponto a ponto. Esse código não quer ser incomodado por pacotes que se perdem pelo caminho. O protocolo de enlace de dados, mostrado no retângulo pontilhado, é o responsável pela confiabilidade das linhas de comunicação, tornando-as perfeitas ou, pelo menos, bastante razoáveis. A propósito, apesar de termos mostrado várias cópias do software da camada de enlace de dados em cada roteador, na verdade, uma única cópia manipula todas as linhas, utilizando diferentes tabelas e estruturas de dados para cada uma delas.

[T3] 3.1.2 Enquadramento

Para oferecer serviços à camada de rede, a camada de enlace de dados deve usar o serviço fornecido a ela pela camada física. O que a camada física faz é aceitar um fluxo de bits brutos e tentar entregá-lo ao destino. Não há uma garantia de que esse fluxo de bits seja livre de erros. O número de bits recebidos pode ser

menor, igual ou maior que o número de bits transmitidos, e eles podem ter valores diferentes dos bits originalmente transmitidos. A camada de enlace de dados é responsável por detectar e, se necessário, corrigir erros.

Em geral, a estratégia adotada pela camada de enlace de dados é dividir o fluxo de bits em quadros e calcular o total de verificação (checksum) em relação a cada quadro. (Os algoritmos de total de verificação serão discutidos mais adiante neste capítulo.) Quando um quadro chega a seu destino, o total de verificação é recalculado. Se o total de verificação recém-calculado for diferente do que está contido no quadro, a camada de enlace de dados saberá que houve um erro e tomará providências para lidar com ele (por exemplo, descartando o quadro defeituoso e possivelmente também enviando de volta um relatório de erros).

A divisão do fluxo de bits em quadros é mais difícil do que parece à primeira vista. Uma forma de obter esse enquadramento é inserir intervalos de tempo entre os quadros, de modo muito semelhante aos espaços entre as palavras de um texto comum. No entanto, as redes raramente oferecem qualquer garantia em relação à temporização. Portanto, é possível que esses intervalos sejam condensados, ou que outros intervalos sejam inseridos durante a transmissão. Como é muito arriscado contar com a temporização para marcar o início e o fim de cada quadro, outros métodos foram criados. Nesta seção, examinaremos quatro métodos:

1. Contagem de caracteres.
2. Bytes de flags, com inserção de bytes.
3. Flags iniciais e finais, com inserção de bits.
4. Violações de codificação da camada física.

O primeiro método de enquadramento utiliza um campo no cabeçalho para especificar o número de caracteres do quadro. Quando vê a contagem de caracteres, a camada de enlace de dados de destino sabe quantos caracteres

devem vir em seguida e, conseqüentemente, onde está o fim do quadro. Essa técnica é mostrada na Figura 3.4(a) para quatro quadros, de tamanhos 5, 5, 8 e 8 caracteres, respectivamente.

[arte: ver original p. 188]

[Dísticos]

[1]Contagem de caracteres Um caractere

[2](a) 5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3

[3] Quadro 1 Quadro 2 Quadro 3 Quadro 4

5 caracteres 5 caracteres 8 caracteres 8 caracteres

[4] Erro

(b) 5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3

[5] Quadro 1 Quadro 2 Agora uma contagem

(Incorreto) de caracteres

[F]Figura 3.4

[FL]Um fluxo de caracteres. (a) Sem erros. (b) Com um erro

O problema com esse algoritmo é que a contagem pode ser adulterada por um erro de transmissão. Por exemplo, se a contagem 5 no segundo quadro da Figura 3.4(b) se tornar 7, o destino perderá a sincronização e não será capaz de localizar o início do quadro seguinte. Mesmo que o total de verificação esteja incorreto, de modo que o destino saiba que o quadro está defeituoso, ele ainda não terá informações suficientes para saber onde começa o quadro seguinte. Enviar um quadro de volta à origem solicitando retransmissão também não ajuda, pois o destino não sabe quantos caracteres deverão ser ignorados para chegar ao início da retransmissão. Por essa razão, o método de contagem de caracteres quase não é mais usado.

O segundo método de enquadramento contorna o problema de ressincronização

após um erro, fazendo cada quadro começar e terminar com bytes especiais. No passado, os bytes iniciais e finais eram diferentes mas, nos últimos anos, a maioria dos protocolos tem utilizado o mesmo byte, chamado **byte de flag**, como delimitador de início e de fim, como mostra a Figura 3.5(a), na qual ele é representado por FLAG. Desse modo, se o receptor perder sincronização, ele poderá simplesmente procurar pelo byte de flag para descobrir o fim do quadro atual. Dois bytes de flag consecutivos indicam o fim de um quadro e o início do próximo.

[arte: ver original p. 189]

[Dísticos]

[1]FLAG Cabeçalho Campo de carga útil Final FLAG

(a)

[2]Caracteres originais

A FLAG B

A ESC B

A ESC FLAG B

A ESC ESC B

[3]Após a inserção

A ESC FLAG B

A ESC ESC B

A ESC ESC ESC FLAG B

A ESC ESC ESC ESC B

(b)

[F]Figura 3.5

[FL](a) Um quadro delimitado por bytes de flag. (b) Quatro exemplos de seqüências de bytes, antes e depois da inserção de bytes

Ocorre um problema sério com esse método quando dados binários, como programas-objeto ou números em ponto flutuante, estão sendo transmitidos. É bem possível que o padrão de bits do byte de flag ocorra nos dados. Em geral, essa situação irá interferir no enquadramento. Uma forma de solucionar esse problema é fazer com que a camada de enlace de dados do transmissor inclua um caractere de escape especial (ESC) imediatamente antes de cada byte de flag "acidental" nos dados. A camada de enlace de dados da extremidade receptora remove o byte de escape antes de entregar os dados à camada de rede. Essa técnica é chamada **inserção de bytes** ou **inserção de caracteres.** Desse modo, é possível distinguir um byte de flag de enquadramento de um byte nos dados pela ausência ou presença de um byte de escape antes dele.

É claro que a próxima pergunta é: o que acontecerá se um byte de escape ocorrer em uma posição intermediária nos dados? Nesse caso, ele também será preenchido com um byte de escape. Desse modo, qualquer byte de escape isolado faz parte de uma seqüência de escape, enquanto o byte de escape duplicado indica que um único escape ocorreu naturalmente nos dados. Alguns exemplos são mostrados na Figura 3-5(b). Em todos os casos, a seqüência de bytes entregue após a remoção dos bytes inseridos é exatamente igual à seqüência de bytes original.

O esquema de inserção de bytes representado na Figura 3.5 é uma ligeira simplificação do que é utilizado no protocolo PPP que a maioria dos computadores domésticos emprega para se comunicar com seu provedor de serviços da Internet. Descreveremos o PPP mais adiante neste capítulo.

Uma das principais desvantagens da utilização desse método de enquadramento é que ele depende da utilização de caracteres de 8 bits. Nem todos os códigos de caracteres utilizam caracteres de 8 bits. Por exemplo, o UNICODE emprega caracteres de 16 bits. À medida que as redes se desenvolveram, as desvantagens

da inclusão do comprimento do código de caracteres no mecanismo de enquadramento se tornaram cada vez mais óbvias; portanto, uma nova técnica teve de ser desenvolvida para permitir o uso de caracteres com tamanhos arbitrários.

A nova técnica permite que os quadros de dados contenham um número arbitrário de bits e possibilita a utilização de códigos de caracteres com um número arbitrário de bits por caractere. De acordo com essa técnica, cada quadro começa e termina com um padrão de bits, 01111110 (na verdade, um byte de flag). Sempre que encontra cinco valores 1 consecutivos nos dados, a camada de enlace de dados do transmissor insere um bit 0 no fluxo de bits que está sendo enviado. Essa **inserção de bits** é semelhante à inserção de bytes, na qual um byte de escape é inserido no fluxo de caracteres enviado antes de ocorrer um byte de flag nos dados.

Ao ver cinco bits 1 consecutivos sendo recebidos, seguidos por um bit 0, o receptor remove automaticamente o bit 0. A inserção de bits, assim como a inserção de bytes, é completamente transparente para a camada de rede de ambos os computadores. Se os dados do usuário contiverem o padrão de flag 01111110, esse flag será transmitido como 011111010, mas será armazenado na memória do receptor como 01111110. A Figura 3.5 mostra um exemplo de inserção de bits.

[arte: ver original p. 190]

[Dísticos]

[1](a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

[2](b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Bits inseridos

[3](c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

[F]Figura 3.5

[FL] Inserção de bits. (a) Os dados originais. (b) Como os dados são exibidos na linha. (c) Como os dados são armazenados na memória do receptor após a remoção de bits

Com a inserção de bits, o limite entre dois quadros pode ser reconhecido sem qualquer tipo de ambigüidade pelo padrão de flags. Desse modo, se o receptor perder o controle de onde estão os dados, bastará varrer a entrada em busca de seqüências de flags, pois elas nunca ocorrem dentro dos dados, apenas nos limites dos quadros.

O último método de enquadramento só se aplica a redes nas quais a decodificação no meio físico contém algum tipo de redundância. Por exemplo, algumas LANs codificam 1 bit de dados utilizando 2 bits físicos. Normalmente, um bit 1 é um par alto-baixo, e um bit 0 é um par baixo-alto. O esquema significa que todo bit de dados tem uma transição intermediária, facilitando a localização dos limites de bits pelo receptor. As combinações alto-alto e baixo-baixo não são usadas para dados, mas são empregadas na delimitação de quadros em alguns protocolos.

Uma observação final a respeito do enquadramento: para proporcionar uma segurança ainda maior, muitos protocolos de enlace de dados utilizam uma combinação de contagem de caracteres com um dos outros métodos. Quando um quadro é recebido, o campo de contagem é utilizado para localizar o fim do quadro. O quadro só é aceito como válido se o delimitador correto estiver presente na posição adequada e se o total de verificação estiver correto. Caso contrário, será preciso examinar o fluxo recebido em busca do delimitador seguinte.

[T3] 3.1.3 Controle de erros

Após resolvermos o problema da delimitação do início e do fim de cada quadro, vamos ao problema seguinte: como ter certeza de que todos os quadros serão entregues na camada de rede de destino, e na ordem apropriada? Suponha que o transmissor simplesmente continue a enviar os quadros sem se importar em saber se eles estão chegando de maneira correta. Essa pode ser uma ótima opção para serviços sem conexão e sem confirmação, mas sem dúvida não seria boa para serviços orientados a conexões confiáveis.

A forma mais comum de garantir uma entrega confiável é dar ao transmissor algum tipo de feedback sobre o que está acontecendo no outro extremo da linha. Normalmente, o protocolo solicita que o receptor retorne quadros de controle especiais com confirmações positivas ou negativas sobre os quadros recebidos. Se receber uma confirmação positiva sobre um quadro, o transmissor saberá que o quadro chegou em segurança ao destino. Por outro lado, uma confirmação negativa significa que algo saiu errado e que o quadro deve ser retransmitido.

Uma complicação adicional decorre da possibilidade de problemas de hardware fazerem com que um quadro desapareça completamente (por exemplo, em uma rajada de ruídos). Nesse caso, o receptor não reagirá de forma alguma, pois não há motivo para isso. Deve ficar claro que um protocolo no qual o transmissor envia um quadro e depois espera por uma confirmação, positiva ou negativa, permanecerá suspenso para sempre caso um quadro tenha sido completamente perdido, por exemplo, em conseqüência de mau funcionamento do hardware.

Essa possibilidade é tratada com a introdução de timers na camada de enlace de dados. Quando o transmissor envia um quadro, em geral ele também inicializa um timer. O timer é ajustado para ser desativado após um intervalo suficientemente longo para o quadro chegar ao destino, ser processado e ter sua confirmação enviada de volta ao transmissor. Em geral, o quadro será recebido de forma correta e a confirmação voltará antes de se alcançar o timeout (tempo

limite) do timer e, nesse caso, o timer será cancelado.

No entanto, se a confirmação ou o quadro se perder, o timer será desativado, alertando o transmissor para um problema potencial. A solução óbvia é simplesmente transmitir o quadro outra vez. Entretanto, quando os quadros são transmitidos várias vezes, existe o perigo de o receptor aceitar o mesmo quadro duas ou mais vezes e de repassá-lo à camada de rede mais de uma vez. Para impedir que isso aconteça, geralmente é necessário atribuir números de seqüência aos quadros enviados, para que o receptor possa distinguir as retransmissões dos quadros originais.

A questão do gerenciamento dos timers e dos números de seqüência para garantir que cada quadro seja realmente passado para a camada de rede do destino exatamente uma vez, nem mais nem menos, é uma parte importante das atribuições da camada de enlace de dados. Mais adiante neste capitulo, estudaremos em detalhes como esse gerenciamento é feito. Para isso, examinaremos uma série de exemplos cada vez mais sofisticados.

[T3] 3.1.4 Controle de fluxo

Outra questão de projeto importante que ocorre na camada de enlace de dados (e também em camadas mais altas) é aquela em que um transmissor quer enviar quadros mais rapidamente do que o receptor é capaz de aceitar. Essa situação pode ocorrer com facilidade quando o transmissor está funcionando em um computador rápido (ou levemente carregado) e o receptor está utilizando um computador lento (ou fortemente carregado). O transmissor fica "bombeando" os quadros em alta velocidade até o receptor ser totalmente "inundado". Mesmo que a transmissão não contenha erros, em um determinado ponto o receptor não será capaz de tratar os quadros à medida que eles chegam e começará a perder alguns deles. Sem dúvida, algo deve ser feito para impedir que essa situação ocorra.

São usadas comumente duas abordagens. Na primeira, chamada **controle de fluxo baseado em feedback**, o receptor envia de volta ao transmissor informações que permitem ao transmissor enviar mais dados, ou que pelo menos mostram ao transmissor qual a situação real do receptor. Na segunda, chamada **controle de fluxo baseado na velocidade**, o protocolo tem um mecanismo interno que limita a velocidade com que os transmissores podem enviar os dados, sem usar o feedback do receptor. Neste capítulo, estudaremos os esquemas de controle de fluxo baseado em feedback, porque os esquemas baseados na velocidade nunca são utilizados na camada de enlace de dados. Examinaremos no Capítulo 5 esquemas baseados na velocidade.

Existem diversos esquemas de controle de fluxo. No entanto, a maioria deles utiliza o mesmo princípio básico. O protocolo contém regras bem definidas sobre quando um transmissor pode enviar o quadro seguinte. Com freqüência, essas regras impedem que os quadros sejam enviados até que o receptor tenha concedido permissão para transmissão, implícita ou explicitamente. Por exemplo, quando uma conexão é estabelecida, o receptor pode informar: "Você está autorizado a me enviar *n* quadros agora, mas depois que eles tiverem sido enviados, não envie mais nada até ser informado de que deve prosseguir." Examinaremos os detalhes em breve.

[T2] 3.2 Detecção e correção de erros

Como vimos no Capítulo 2, o sistema telefônico tem três partes: os switches, os troncos entre estações (troncos interurbanos) e os loops locais. Os dois primeiros são agora quase inteiramente digitais na maioria dos países desenvolvidos. Os loops locais ainda são pares de fios de cobre trançados analógicos, e continuarão assim durante anos, devido ao elevado custo de sua substituição. Embora os erros sejam raros na parte digital, eles ainda são comuns nos loops locais. Além

disso, a comunicação sem fio está se tornando mais comum, e as taxas de erros nesse caso são várias ordens de grandeza piores do que as taxas de erros dos troncos interurbanos de fibra óptica. A conclusão é que os erros de transmissão ainda estarão presentes por muitos anos. Teremos de aprender a lidar com eles. Como resultado dos processos físicos que os geram, os erros em alguns meios (por exemplo, o rádio) tendem a ocorrer com mais freqüência em grandes volumes (rajadas) do que isoladamente. O fato de os erros acontecerem em rajadas tem vantagens e desvantagens em relação aos erros isolados de um único bit. Uma vantagem é que os dados de computadores são sempre enviados em blocos de bits. Suponha que o tamanho do bloco seja 1000 bits e que a taxa de erros seja 0,001 por bit. Porém, se os erros surgirem em rajadas de 100, apenas um ou dois blocos em 100 será(ão) afetado(s), em média. A desvantagem dos erros em rajada é que eles são muito mais difíceis de corrigir que os erros isolados.

[T3] 3.2.1 Códigos de correção de erros

Os projetistas de redes desenvolveram duas estratégias básicas para tratar os erros. Uma delas é incluir informações redundantes suficientes em cada bloco de dados enviado. Com isso, o receptor é capaz de deduzir quais devem ter sido os dados transmitidos. A outra forma é incluir uma redundância suficiente apenas para permitir que o receptor deduza que houve um erro, mas sem identificar qual, e solicite uma retransmissão. A primeira estratégia utiliza **códigos de correção de erros**, e a outra emprega **códigos de detecção de erros**. O uso de códigos de correção de erros freqüentemente é denominado **correção antecipada de erros**.

Cada uma dessas técnicas ocupa um nicho ecológico diferente. Em canais altamente confiáveis, como os de fibra, é mais econômico utilizar um código de

detecção de erros e simplesmente retransmitir o bloco defeituoso ocasional. Porém, em canais como enlaces sem fio que geram muitos erros, é melhor adicionar a cada bloco redundância suficiente para que o receptor seja capaz de descobrir qual era o bloco original, em vez de confiar em uma retransmissão, que pode ela própria conter erros.

Para entender como os erros podem ser tratados, é necessário verificar de perto o que é de fato um erro. Normalmente, um quadro consiste em m bits de dados (ou seja, de mensagens) e de r bits redundantes ou de verificação. Seja o tamanho total n (isto é, $n = m + r$). Com freqüência, uma unidade de n bits que contém bits de dados e bits de verificação é chamada **palavra de código** (codeword) de n bits.

Dadas duas palavras de código, digamos 10001001 e 10110001, é possível determinar quantos bits correspondentes apresentam diferenças. Nesse caso, são 3 os bits divergentes. Para determinar quantos bits apresentam diferenças, basta efetuar uma operação OR exclusivo entre as duas palavras de código, e contar o número de bits 1 no resultado. Por exemplo:

[TD]

10001001

<u>10110001</u>

00111000 [TN]

O número de posições de bits em que duas palavras de código diferem entre si é chamado **distância de Hamming** (Hamming, 1950). Isso significa que, se duas palavras de código estiverem a uma distância de Hamming igual a d uma a outra, será necessário corrigir d erros de bits isolados para converter uma palavra na outra.

Na maioria das aplicações de transmissão de dados, todas as 2^m mensagens de dados possíveis são válidas; no entanto, devido à forma como os bits de

verificação são calculados, nem todas as 2^n palavras de código possíveis são usadas. Dado o algoritmo para cálculo dos bits de verificação, é possível elaborar uma lista completa contendo as palavras de código válidas. A partir dessa lista, é possível localizar as duas palavras de código cuja distância de Hamming é mínima. Essa distância é a distância de Hamming do código completo.

As propriedades de detecção e de correção de erros de um código dependem de sua distância de Hamming. Para detectar *d* erros, você precisa de um código de distância $d + 1$, pois com tal código não há como *d* erros de bits transformarem uma palavra de código válida em outra palavra de código válida. Ao detectar uma palavra de código inválida, o receptor poderá perceber que houve um erro de transmissão. Da mesma forma, para corrigir *d* erros, você precisa de um código de distância $2d + 1$ porque, dessa forma, as palavras de código válidas estarão tão distantes que, mesmo com *d* alterações, a palavra de código original continuará mais próxima do que qualquer outra e poderá ser determinada univocamente.

Como um exemplo simples de código de detecção de erros, imagine um código no qual um único **bit de paridade** é acrescentado aos dados. O bit de paridade é escolhido de forma que o número de bits 1 da palavra de código seja par (ou ímpar). Por exemplo, quando 1011010 é enviado com paridade par, é acrescentado um bit ao final para formar 10110100. Com paridade ímpar, 1011010 passa a ser 10110101. Um código com um único bit de paridade tem uma distância igual a 2, pois qualquer erro de um único bit produz uma palavra de código com a paridade errada. Isso pode ser usado para detectar erros isolados.

Como um exemplo simples de código de correção de erros, considere um código contendo apenas quatro palavras de código válidas:

0000000000, 0000011111, 1111100000 e 1111111111

Esse código tem uma distância igual a 5, o que significa que ele pode corrigir erros duplos. Se a palavra de código 0000000111 for detectada, o receptor saberá que a original deve ter sido 0000011111. No entanto, se um erro triplo transformar 0000000000 em 0000000111, o erro não será corrigido da maneira adequada.

Suponha que desejamos criar um código com m bits de mensagem e r bits de verificação que permitirão a correção de todos os erros simples. Cada uma das 2^m mensagens válidas tem n palavras de código inválidas a uma distância igual a 1 da mensagem. Essas palavras inválidas são formadas pela inversão sistemática de cada um dos n bits da palavra de código de n bits formada a partir dela. Portanto, cada uma das 2^m mensagens válidas exige $n + 1$ padrões de bits dedicados a ela. Como o número total de padrões de bits é 2^n, devemos ter $(n + 1)2^m \leq 2^n$. Utilizando $n = m + r$, esse requisito passa a ser $(m + r + 1) \leq 2^r$. Se m for determinado, o limite para o número de bits de verificação necessários para corrigir erros isolados será mais baixo.

Esse limite teórico mais baixo pode, na verdade, ser alcançado pela utilização de um método criado por Hamming (1950). Os bits da palavra de código são numerados consecutivamente, começando com o bit 1 da extremidade esquerda, com o bit 2 imediatamente à sua direita e assim por diante. Os bits que são potências de 2 (1, 2, 4, 8, 16 etc.) são bits de verificação. Os outros (3, 5, 6, 7, 9 etc.) são preenchidos com os m bits de dados. Cada bit de verificação força a paridade de algum conjunto de bits, incluindo seu próprio conjunto, a ser par (ou ímpar). Um bit pode ser incluído em vários cálculos de paridade. Se quiser ver para quais bits de verificação o bit de dados da posição k contribui, reescreva k como a soma de potências de 2. Por exemplo, $11 = 1 + 2 + 8$ e $29 = 1 + 4 + 8 + 16$. Um bit é verificado apenas pelos bits de verificação que ocorrem em sua expansão (por exemplo, o bit 11 é verificado pelos bits 1, 2 e 8).

Quando uma palavra de código é recebida, o receptor inicializa um contador como zero. Em seguida, ele examina cada bit de verificação *k* (*k* = 1, 2, 4, 8, ...) para confirmar se a paridade está correta. Caso não esteja, *k* é incluído no contador. Se o contador indicar zero após todos os bits de verificação terem sido examinados (ou seja, se todos estiverem corretos), a palavra de código será aceita como válida. Se o contador não for igual a zero, ele conterá o número do bit incorreto. Por exemplo, se os bits de verificação 1, 2 e 8 estiverem incorretos, o bit invertido será igual a 11, pois ele é o único verificado pelos bits 1, 2 e 8. A Figura 3.7 mostra alguns caracteres ASCII de 7 bits codificados como palavras de código de 11 bits, pela utilização de um código de Hamming. Lembre-se de que os dados são encontrados nas posições de bits 3, 5, 6, 7, 9, 10 e 11.

[arte: ver original p. 195]

[Tabela]

Caractere	ASCII	Bits de verificação
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Ordem de transmissão de bits

[F]Figura 3.7

[FL] Utilização de um código de Hamming para corrigir erros em rajada

Os códigos de Hamming só são capazes de corrigir erros simples. No entanto, existe um artifício que pode ser usado para permitir que códigos de Hamming corrijam erros em rajada. Uma seqüência de *k* palavras consecutivas é organizada como uma matriz, com uma palavra de código por linha. Normalmente, os dados seriam transmitidos uma palavra de código por vez, da esquerda para a direita. Para corrigir erros em rajada, os dados devem ser transmitidos uma coluna de cada vez, começando pela coluna mais à esquerda. Quando todos os *k* bits tiverem sido enviados, uma segunda coluna será enviada e assim por diante, como indica a Figura 3.7. Quando o quadro chegar ao receptor, a matriz será reconstruída, uma coluna de cada vez. Se ocorrer um erro em rajada com a extensão *k*, no máximo 1 bit de cada uma das *k* palavras de código será afetado, mas o código de Hamming poderá corrigir um erro por palavra de código, possibilitando a restauração do bloco inteiro. Esse método utiliza *kr* bits de verificação para tornar blocos de *km* bits de dados imunes a um único erro em rajada que tenha uma extensão menor ou igual a *k*.

[T3] 3.2.2 Códigos de detecção de erros

Os códigos de correção de erros são extensamente utilizados em enlaces sem fios, conhecidos por serem ruidosos e propensos a erros em comparação com a fiação de cobre ou a fibra óptica. Sem códigos de correção de erros, seria difícil conseguir algo. Porém, usando-se fio de cobre ou fibra, a taxa de erros é muito mais baixa, e assim a detecção de erros e retransmissão em geral é mais eficiente para lidar com o erro ocasional.

Como exemplo, considere um canal no qual os erros são isolados e a taxa de

erros é 10^{-6} por bit. Defina o tamanho do bloco como 1.000 bits. Para proporcionar a correção de erros de blocos de 1.000 bits, são necessários 10 bits de verificação; um megabit de dados necessitaria de 10.000 bits de verificação. Para detectar um bloco com um erro simples de 1 bit, um bit de paridade por bloco seria suficiente. A cada 1.000 blocos, um bloco extra terá de ser transmitido (totalizando 1.001 bits). O overhead total para o método de detecção de erros + retransmissão é de apenas 2.001 bits por megabit de dados, contra 10.000 bits para um código de Hamming.

Se um único bit de paridade for incluído em um bloco e o bloco for seriamente adulterado por um longo erro em rajada, a probabilidade de que o erro seja detectado é de apenas 0,5, o que não é muito aceitável. As disparidades poderão ser consideravelmente melhoradas se cada bloco for enviado como uma matriz retangular com *n* bits de largura e *k* bits de altura, conforme descrevemos. Um bit de paridade é calculado separadamente para cada coluna e afixado à matriz como sua última linha. Em seguida, a matriz é transmitida uma linha de cada vez. Quando o bloco chega a seu destino, o receptor verifica todos os bits de paridade. Se um deles estiver errado, será solicitada uma nova transmissão do bloco. Além disso, serão solicitadas retransmissões adicionais de acordo com a necessidade, até um bloco inteiro ser recebido sem quaisquer erros de paridade. Esse método é capaz de detectar uma única rajada de tamanho *n*, pois será alterado apenas 1 bit por coluna. No entanto, uma rajada de tamanho $n + 1$ não será detectada se o primeiro bit estiver invertido, o último bit estiver invertido e todos os outros estiverem corretos. (Um erro em rajada não implica que todos os bits estejam errados; na verdade, ele significa que pelo menos o primeiro e o último bits estão errados.) Se o bloco for seriamente adulterado por uma longa rajada ou por várias rajadas mais curtas, a probabilidade de que qualquer uma das *n* colunas tenha a paridade correta por acidente é de 0,5. Portanto, a

probabilidade de um bloco defeituoso ser aceito quando não deveria sê-lo é de 2^n.

Embora às vezes esse esquema se mostre adequado, na prática outro método teve seu uso mais difundido: o **código polinomial**, também conhecido como **código de redundância cíclica** ou **CRC (Cyclic Redundancy Check).** Os códigos polinomiais se baseiam no tratamento de strings de bits como representações de polinômios com coeficientes 0 e 1 apenas. Um quadro de k bits é considerado a lista de coeficientes para um polinômio com k termos, variando desde x^{k-1} até a x^0. Dizemos que tal polinômio é de grau $k - 1$. O bit de alta ordem (mais à esquerda) é o coeficiente de x^{k-1}; o bit seguinte é o coeficiente de x^{k-2} e assim por diante. Por exemplo, 110001 tem 6 bits, portanto representa um polinômio de seis termos com os coeficientes 1, 1, 0, 0, 0 e 1: $x^5 + x^4 + x^0$.

A aritmética polinomial é feita em módulo 2, de acordo com as regras da teoria algébrica. Não há transportes para a adição nem empréstimos para a subtração. Tanto a adição quanto a subtração são idênticas à operação OR exclusivo.

Considere o seguinte exemplo:

```
  10011011      00110011      11110000      01010101
+ 11001010    + 11001101    – 10100110    – 10101111
  01010001      11111110      01010110      11111010
```

A divisão longa é efetuada do mesmo modo que em binário, exceto pelo fato de a subtração ser de módulo 2, como mostramos anteriormente. Diz-se que um divisor "cabe em" um dividendo se o dividendo tem a mesma quantidade de bits do divisor.

Quando o método do código polinomial é empregado, o transmissor e o receptor devem concordar em relação a um **polinômio gerador**, $G(x)$, antecipadamente. Tanto o bit de mais alta ordem quanto o de mais baixa ordem do polinômio gerador devem ser iguais a 1. Para calcular o **total de verificação** (checksum) de

um quadro com m bits, que corresponde ao polinômio $M(x)$, o quadro deve ter mais bits do que o polinômio gerador. A idéia é acrescentar um total de verificação ao final do quadro, de forma que o polinômio representado pelo quadro verificado pela soma seja divisível por $G(x)$. Quando obtiver o quadro verificado, o receptor tentará dividi-lo por $G(x)$. A existência de um resto indica que houve um erro de transmissão.

O algoritmo para calcular o total de verificação é:

1. Seja r o grau de $G(x)$. Acrescente r bits zero à extremidade de baixa ordem do quadro, de modo que ele passe a conter $m + r$ bits e corresponda ao polinômio $x^rM(x)$.

2. Divida o string de bits correspondente a $G(x)$ pelo string de bits correspondente a $x^rM(x)$ utilizando a divisão de módulo 2.

3. Subtraia o resto (que tem sempre r ou menos bits) do string de bits correspondente a $x^rM(x)$ utilizando a subtração de módulo 2.

O resultado é o quadro verificado pela soma que deverá ser transmitido. Chame o polinômio de $T(x)$.

A Figura 3.8 ilustra o cálculo referente a um quadro 1101011011, usando o gerador $G(x) = x^4 + x + 1$.

O polinômio $T(x)$ deverá ser divisível (em módulo 2) por $G(x)$. Em qualquer problema de divisão, se você subtrair o resto do dividendo, o resultado será divisível pelo divisor. Por exemplo, na base 10, se você dividir 210.278 por 10.941, o resto será 2.399. Subtraindo-se 2.399 de 210.278, o resultado final (207.879) será divisível por 10.941.

Agora vamos analisar a abrangência desse método. Que tipos de erros serão detectados? Imagine que ocorra um erro de transmissão, de forma que, em lugar de chegar o string de bits correspondente a $T(x)$, seja recebida a soma $T(x) + E(x)$. Cada bit 1 de $E(x)$ corresponde a um bit que foi invertido. Se houver k bits 1

em $E(x)$, isso significa que ocorreram k erros de bits simples. Um único erro em rajada é caracterizado por um bit 1 inicial, uma mistura de bits 0 e 1 e um bit 1 final, sendo todos os outros bits iguais a 0.

[arte: ver original p. 198]

[Dísticos]

[1]Quadro: 1 1 0 1 0 1 1 0 1 1

Gerador: 1 0 0 1 1

Mensagem após o acréscimo de 4 bits zero: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

[2] 1 1 0 0 0 0 1 0 1 0

1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 0 0 0 0

1 0 0 1 1

1 0 0 1 1

1 0 0 1 1

0 0 0 0 1

0 0 0 0 0

0 0 0 1 0

0 0 0 0 0

0 0 1 0 1

0 0 0 0 0

0 1 0 1 1

0 0 0 0 0

1 0 1 1 0

1 0 0 1 1

0 1 0 1 0

0 0 0 0 0

1 0 1 0 0

1 0 0 1 1

0 1 1 1 0

0 0 0 0 0 Resto

1 1 1 0

[3]Quadro transmitido: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

[F]Figura 3.8

[FL] Cálculo do total de verificação do código polinomial

Ao receber o quadro com total de verificação, o receptor o divide por $G(x)$; ou seja, ele calcula $[T(x) + E(x)]/G(x)$. $T(x)/G(x)$ é igual a 0; portanto, o resultado do cálculo é simplesmente $E(x)/G(x)$. Os erros que corresponderem a polinômios contendo $G(x)$ como fator serão simplesmente ignorados; todos os outros erros serão detectados.

Se houver ocorrido um erro de um único bit, $E(x) = x^i$, onde i determina o bit incorreto. Se contiver dois ou mais termos, $G(x)$ nunca dividirá $E(x)$; portanto, todos os erros de um único bit serão detectados.

Se tiverem ocorrido dois erros isolados de um único bit, $E(x) = x^i + x^j$, onde $i > j$. Como alternativa, esse cálculo pode ser representado por $E(x) = x^j (x^{i-j} + 1)$. Se supusermos que $G(x)$ não é divisível por x, uma condição suficiente para todos os erros duplos serem detectados é $G(x)$ não dividir $x^k + 1$ para qualquer k até o valor máximo de $i - j$ (isto é, até o comprimento máximo do quadro). São conhecidos polinômios simples de grau baixo que protegem quadros longos. Por exemplo, $x^{15} + x^{14} + 1$ não dividirá $x^k + 1$ para qualquer valor de k abaixo de 32.768.

Se houver um número ímpar de bits com erros, $E(x)$ conterá um número ímpar de termos (por exemplo, $x^5 + x^2 + 1$, mas não $x^2 + 1$). É interessante observar que nenhum polinômio com um número ímpar de termos terá $x + 1$ como fator no sistema de módulo 2. Ao tornar $x + 1$ um fator de $G(x)$, podemos detectar todos

os erros que consistem em um número ímpar de bits invertidos.

Para confirmar que nenhum polinômio com um número ímpar de termos será divisível por $x + 1$, suponha que $E(x)$ tenha um número ímpar de termos e seja divisível por $x + 1$. Fatore $E(x)$ em $(x + 1)\ Q(x)$. Agora, avalie $E(1) = (1 + 1)Q(1)$. Como $1 + 1 = 0$ (em módulo 2), $E(1)$ deve ser igual a zero. Se $E(x)$ tiver um número ímpar de termos, a utilização de 1 no lugar de x sempre produzirá 1 como resultado. Portanto, nenhum polinômio com um número ímpar de termos será divisível por $x + 1$.

Por último, e mais importante, um código polinomial com r bits de verificação detectará todos os erros em rajada que tiverem um tamanho $\leq r$. Um erro em rajada de tamanho k pode ser representado por $x^i(x^{k-1} + \ldots + 1)$, onde i determina a distância entre a rajada e a extremidade direita do quadro recebido. Se contiver um termo x^0, $G(x)$ não terá x^i como fator; portanto, se o grau da expressão entre parênteses for menor que o grau de $G(x)$, o resto nunca poderá ser igual a zero.

Se o tamanho da rajada for $r + 1$, o restante da divisão por $G(x)$ será zero se e somente se a rajada for idêntica a $G(x)$. Por definição de rajada, o primeiro e o último bits de uma rajada devem ser iguais a 1; assim, a correspondência entre os valores dependerá dos $r - 1$ bits intermediários. Se todas as combinações forem consideradas igualmente prováveis, a probabilidade desse quadro incorreto ser aceito como válido será de $½^{r-1}$.

Também podemos mostrar que, ao ocorrer um erro em rajada com mais de $r + 1$ bits ou forem registradas várias rajadas mais curtas, a probabilidade de um quadro defeituoso passar sem ser percebido poderá ser igual a $½^r$, supondo-se que todos os padrões de bits sejam igualmente prováveis.

Certos polinômios se tornaram padrões internacionais. O que é utilizado no IEEE 802 é:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Entre outras características interessantes, ele tem a propriedade de detectar todas as rajadas de comprimento 32 ou menos e todos as rajadas que afetam um número ímpar de bits.

Apesar do cálculo necessário para computar o total de verificação parecer complicado, Peterson e Brown (1961) mostraram que é possível criar um simples circuito shift register (de registrador de deslocamento) para calcular e conferir os totais de verificação em hardware. Na prática, esse hardware quase sempre é utilizado. Virtualmente todas as LANs o empregam, como também as linhas ponto a ponto o utilizam em alguns casos.

Durante décadas, imaginou-se que todos os quadros a serem verificados por meio do total de verificação continham bits aleatórios. Todas as análises de algoritmos de total de verificação foram realizadas com base nessa premissa. Uma inspeção de dados reais mostrou que essa suposição estava bastante errada. Percebeu-se, por exemplo, que sob determinadas circunstâncias, os erros não detectados são muito mais comuns do que se supunha (Partridge *et al.*, 1995).

[T2] 3.3 Protocolos elementares de enlace de dados

Como uma introdução ao estudo dos protocolos, vamos começar examinando três protocolos com grau de complexidade crescente. Para os leitores interessados, há um simulador disponível para esses e outros protocolos na Web (veja o prefácio). Antes de examinarmos os protocolos, é útil tornar explícitas algumas das suposições nas quais se baseia o modelo de comunicação. Para começar, supomos que, na camada física, na camada de enlace de dados e na camada de rede existem processos independentes que se comunicam pelo envio de mensagens. Em muitos casos, os processos da camada física e da camada de enlace de dados estarão funcionando em um processador dentro de um chip

especial de E/S de rede, e o código da camada de rede estará na CPU principal. Porém, outras implementações também são possíveis (por exemplo, três processos em um único chip de E/S, ou as camadas física e de enlace de dados funcionando como procedimentos chamados pelo processo da camada de rede). De qualquer forma, tratar as três camadas como processos separados torna a discussão conceitualmente mais clara e também enfatiza a independência das camadas.

Outra suposição de extrema importância é a de que a máquina *A* deseja enviar um longo fluxo de dados à máquina *B* utilizando um serviço confiável, orientado a conexões. Mais adiante, vamos considerar a situação em que *B* também deseja enviar os dados a *A* simultaneamente. Supõe-se que *A* tem um suprimento infinito de dados prontos para serem enviados, e nunca terá de esperar pela produção de dados. Quando a camada de dados de *A* solicitar dados, a camada de rede sempre será capaz de obedecer de imediato. (Mais adiante essa restrição também será superada.)

Também supomos que as máquinas não sofrerão panes. Isto é, esses protocolos lidam com erros de comunicação, mas não com os problemas causados por computadores que sofrem panes e são reinicializados.

No que se refere à camada de enlace de dados, o pacote repassado a ela pela camada de rede através da interface consiste em dados puros, em que cada bit deve ser entregue à camada de rede de destino. O fato de a camada de rede de destino poder interpretar parte do pacote como um cabeçalho não tem nenhum interesse para a camada de enlace de dados.

Quando a camada de enlace de dados aceita um pacote, ela o encapsula em um quadro, acrescentando-lhe um cabeçalho e um final de enlace de dados (veja a Figura 3.1). Portanto, um quadro consiste em um pacote incorporado, em algumas informações de controle (no cabeçalho) e em um total de verificação (no

final). Em seguida, o quadro é transmitido à camada de enlace de dados da outra máquina. Presumiremos que existem procedimentos de biblioteca adequados, *to_physical_layer* para enviar um quadro e *from_physical_layer* para receber um quadro. O hardware de transmissão calcula e acrescenta o total de verificação (criando assim o final), de forma que o software da camada de enlace de dados não precise se preocupar com isso. Por exemplo, poderia ser usado o algoritmo polinomial discutido anteriormente neste capítulo.

Inicialmente, o receptor nada tem a fazer. Ele fica à espera de que algo aconteça. Nos exemplos de protocolos apresentados neste capítulo, indicaremos que a camada de enlace de dados está esperando que algo aconteça por meio de chamada de procedimento *wait_for_event*(*&event*). Esse procedimento só retorna quando acontece algo (por exemplo, quando chega um quadro). Ao retornar, a variável *event* informa o que aconteceu. O conjunto de eventos possíveis é diferente para os diversos protocolos a serem descritos e será definido separadamente para cada protocolo. Observe que, em uma situação mais realista, a camada de enlace de dados não ficará em um loop estrito à espera de um evento, como sugerimos, mas receberá uma interrupção, o que a fará interromper o que estava fazendo para manipular o quadro recebido. Apesar disso, por simplicidade, ignoraremos todos os detalhes de atividades paralelas na camada de enlace de dados, e presumiremos que ela se dedica em tempo integral apenas ao tratamento do nosso canal.

Quando um quadro chega ao receptor, o hardware calcula o total de verificação. Se o total de verificação estiver incorreto (ou seja, se houve um erro de transmissão), a camada de enlace de dados será informada (*event* = *cksum_err*). Se o quadro recebido tiver chegado sem danos, a camada de enlace de dados também será informada (*event* = *frame_arrival*), para que ela possa receber o quadro para inspeção usando *from_physical_layer*. Assim que recebe um quadro

correto, a camada de enlace de dados verifica as informações de controle contidas no cabeçalho e, se tudo estiver correto, repassa a porção do pacote à camada de rede. Em nenhuma circunstância, um cabeçalho de quadro será entregue a uma camada de rede.

Há uma boa razão para que a camada de rede nunca receba qualquer parte do quadro: manter os protocolos de rede e de enlace de dados completamente separados. Desde que a camada de rede não saiba absolutamente nada sobre o protocolo de enlace de dados ou sobre o formato do quadro, esses itens poderão ser alterados sem exigir mudanças no software da camada de rede. A utilização de uma interface rígida entre a camada de rede e a camada de enlace de dados simplifica bastante o projeto do software, porque os protocolos de comunicação das diferentes camadas podem evoluir de forma independente.

A Figura 3.9 mostra algumas declarações (em linguagem C) comuns a muitos dos protocolos que serão discutidos mais adiante. Cinco estruturas de dados são definidas nessa figura: *boolean*, *seq_nr*, *packet*, *frame_kind* e *frame*. Um *boolean* é um tipo enumerado e pode assumir os valores *verdadeiro* (*true*) e *falso* (*false*). Um *seq_nr* é um inteiro pequeno usado para numerar os quadros, o que facilita sua distinção. Esses números de seqüência variam de 0 até *MAX_SEQ* (inclusive), que representa um limite a ser definido, quando necessário, para cada protocolo. Um *packet* é a unidade de informação trocada entre a camada de rede e a camada de enlace de dados da mesma máquina, ou entre pares da camada de rede. No nosso modelo, ele sempre contém *MAX_PKT* bytes; no entanto, de modo mais realista, ele teria comprimento variável.

Um *frame* é composto por quatro campos: *kind*, *seq*, *ack* e *info*; os três primeiros contêm informações de controle, e o último contém os dados reais a serem transferidos. Esses campos de controle são chamados coletivamente **cabeçalho de quadro**.

[arte: ver original da p. 202]

[TD]

```
#define MAX_PKT 1024                          /* determina tamanho
máximo de pacote em bytes */

typedef enum {false, true} boolean;           /* tipo booleano */
typedef unsigned int seq_nr;                  /* números de confirmação ou
seqüência */
typedef struct {unsigned char data[MAX_PKT];} packet;    /* definição de pacote
*/
typedef enum {data, ack, nak} frame_kind;     /* definição de frame_kind */

typedef struct {                              /* quadros são transportados nessa
camada */
  frame_kind kind;                            /* que tipo de quadro é este? */
  seq_nr seq;                                 /* número de seqüência */
  seq_nr ack;                                 /* número de confirmação */
  packet info;                                /* o pacote da camada de rede */
} frame;

/* Espera que um evento aconteça; retorna o tipo em event. */
void wait_for_event(event_type *event);

/* Busca um pacote da camada de rede para transmissão no canal. */
void from_network_layer(packet *p);

/* Entrega informações de um quadro recebido à camada de rede. */
```

```
void to_network_layer(packet *p);


/* Extrai um quadro recebido da camada física e o copia em r. */

void from_physical_layer(frame *r);


/* Repassa o quadro à camada física para transmissão. */

void to_physical_layer(frame *s);


/* Inicializa o relógio e ativa o evento timeout. */

void start_timer(seq_nr k);


/* Interrompe o relógio e desativa o evento timeout. */

void stop_timer(seq_nr k);


/* Inicializa um timer auxiliar e ativa o evento ack_timeout. */

void start_ack_timer(void);


/* Interrompe o timer auxiliar e desativa o evento ack_timeout. */

void stop_ack_timer(void);


/* Permite que a camada de rede dê inicio a um evento network_layer_ready. */

void enable_network_layer(void);


/* Impede que a camada de rede dê início a um evento network_layer_ready. */

void disable_network_layer(void);


/* A macro inc é expandida em linha: incrementa k de forma circular. */
```

```
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0 [TN]
```

[F]Figura 3.9

[FL] Algumas definições utilizadas nos protocolos apresentados a seguir. Essas definições estão armazenadas no arquivo *protocol.h*

O campo *kind* indica se há dados no quadro, pois alguns protocolos distinguem quadros que contêm exclusivamente informações de controle daqueles que armazenam dados além dessas informações. Os campos *seq* e *ack* são usados para números de seqüência e confirmações, respectivamente; seu uso será descrito detalhamente mais adiante. O campo *info* de um quadro de dados contém um único pacote; o campo *info* de um quadro de controle não é usado. Uma implementação mais realista utilizaria um campo *info* de comprimento variável; nos quadros de controle, esse campo seria completamente omitido.

É importante compreender o relacionamento entre um pacote e um quadro. A camada de rede cria um pacote tomando uma mensagem da camada de transporte e acrescentando a ela o cabeçalho da camada de rede. Esse pacote *é* repassado à camada de enlace de dados para inclusão no campo *info* de um quadro que esteja sendo enviado. Quando o quadro chega ao destino, a camada de enlace de dados extrai o pacote do quadro e envia o pacote à camada de rede. Dessa forma, a camada de rede pode atuar com se as máquinas pudessem trocar pacotes diretamente.

Na Figura 3.9 também estão listados diversos procedimentos. Esses procedimentos são rotinas de biblioteca cujos detalhes são dependentes da implementação e cujo funcionamento interno não será discutido aqui. O procedimento *wait_for_event* permanece à espera de que algo aconteça, como mencionamos anteriormente. Os procedimentos *to_network_layer* e *from_network_layer* são usados pela camada de enlace de dados para enviar

pacotes à camada de rede e aceitar pacotes da camada de rede, respectivamente. Observe que *from_physical_layer* e *to_physical_layer* repassam quadros entre a camada de enlace de dados e a camada física. Por outro lado, os procedimentos *to_network_layer* e *from_network_layer* repassam pacotes entre a camada de enlace de dados e a camada de rede. Em outras palavras, *to_network_layer* e *from_network_layer* lidam com a interface entre as camadas 2 e 3, enquanto *from_physical_layer* e *to_physical_layer* lidam com a interface entre as camadas 1 e 2.

Na maioria dos protocolos, supomos o uso de um canal não confiável que perde quadros inteiros ocasionalmente. Para se recuperar dessas calamidades, a camada de enlace de dados transmissora tem de inicializar um timer ou relógio interno, sempre que envia um quadro. Se nenhuma confirmação tiver sido recebida dentro de um intervalo de tempo predeterminado, o relógio chegará ao timeout e a camada de enlace de dados receberá um sinal de interrupção.

Em nossos protocolos, isso é tratado permitindo-se ao procedimento *wait_for_event* retornar *event* = *timeout*. Os procedimentos *start_timer* e *stop_timer* ativam e desativam o timer, respectivamente. Os timeouts só são possíveis quando o timer está funcionando. É explicitamente permitido chamar *start_timer* enquanto o timer está funcionando; essa chamada simplesmente reinicializa o relógio para provocar o próximo timeout, depois de decorrer um intervalo de timer (a menos que ele seja reinicializado ou desativado durante esse intervalo).

Os procedimentos *start_ack_timer* e *stop_ack_timer* controlam um timer auxiliar cuja função é gerar confirmações sob determinadas condições.

Os procedimentos *enable_network_layer* e *disable_network_layer* são usados nos protocolos mais sofisticados, para os quais não mais supomos que a camada de rede sempre terá pacotes a serem enviados. Quando a camada de enlace de

dados habilita a camada de rede, esta passa a ter permissão para causar uma interrupção sempre que tiver um pacote para enviar. Isso é indicado por *event = network_layer_ready*. Quando uma camada de rede está inativa, ela não pode causar tais eventos. Definindo com cuidado os momentos em que ativa e desativa a camada de rede, a camada de enlace de dados pode impedir que a camada de rede acabe ficando sobrecarregada com pacotes para os quais não dispõe de espaço no buffer.

Os números de seqüência dos quadros estão sempre na faixa de 0 a *MAX_SEQ* (inclusive), onde *MAX_SEQ* tem um valor diferente para os diversos protocolos. Com freqüência, é necessário aumentar um número de seqüência em uma unidade, de forma circular (isto é, *MAX_SEQ* é seguido por 0). A macro *inc* cuida dessa incrementação. Ela é definida como uma macro porque é usada em linha no caminho crítico. Como veremos mais adiante, com freqüência o processamento de protocolos é o fator que limita o desempenho da rede; portanto, a definição de operações simples como macros não afeta a legibilidade do código, mas melhora o desempenho. Além disso, como *MAX_SEQ* passa a ter diferentes valores em diferentes protocolos ao ser transformado em uma macro, é possível incluir todos os protocolos no mesmo código binário sem que haja conflito. Essa possibilidade é muito útil para o simulador.

As declarações da Figura 3.9 fazem parte de cada um dos protocolos apresentados a seguir. Para economizar espaço e facilitar a consulta, essas declarações foram extraídas dos protocolos e são apresentadas todas juntas, mas conceitualmente elas devem estar integradas aos protocolos. Na linguagem C, essa integração é feita inserindo-se as definições em um arquivo de cabeçalho especial, nesse caso *protocol.h*, e utilizando-se o recurso #include do pré-processador C, que inclui essas definições nos arquivos de protocolo.

[T3] 3.3.1 Um protocolo simplex sem restrições

Como primeiro exemplo, consideraremos um protocolo muito simples. Os dados são transmitidos apenas em um sentido. As camadas de rede do transmissor e do receptor estão sempre prontas à espera de informações. O tempo de processamento pode ser ignorado. O espaço disponível em buffer é infinito. E o melhor de tudo é que o canal de comunicação entre as camadas de enlace de dados nunca é danificado nem perde quadros. Esse protocolo absolutamente imaginário, que denominaremos "utopia", é mostrado na Figura 3.10.

O protocolo consiste em dois procedimentos distintos, um que envia e outro que recebe informações. O procedimento transmissor é executado na camada de enlace de dados da máquina de origem, e o receptor é executado na camada de enlace de dados da máquina de destino. Não são usados números de seqüência ou de confirmação; portanto, *MAX_SEQ* não é necessário. O único tipo de evento possível é *frame_arrival* (ou seja, a chegada de um quadro não danificado).

O transmissor é um loop [TD]whi1e[TN] infinito que envia os dados o mais rápido possível. O corpo do loop é formado por três ações: buscar um pacote da (sempre prestativa) camada de rede, criar um quadro utilizando a variável *s* e transmitir o quadro ao destino. Apenas o campo *info* do quadro é usado por esse protocolo, pois os outros campos se referem ao controle de fluxo e de erros e, nesse caso, não há erros nem restrições de controle de fluxo.

[arte: ver original da p. 205]

[TD]

/* O protocolo 1 (utopia) oferece transmissão de dados em um único sentido, do transmissor para o receptor. Pressupõe-se que o canal de comunicação é livre de erros e que o receptor é capaz de processar toda a entrada de uma forma infinitamente rápida. Conseqüentemente, o transmissor permanece em um loop enviando os dados com a maior rapidez possível. */

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"


void sender1(void)
{
  frame s;                                  /* buffer para um quadro enviado */
  packet buffer;                            /* buffer para um pacote enviado */


  while (true) {
        from_network_layer(&buffer);        /* obtém algo para enviar */
        s.info = buffer;                    /* copia em s para transmitir */
        to_physical_layer(&s);              /* envia ao destino */
  }                                         /* O amanhã, o amanhã, o amanhã
                                               avança nesse passo pequeno, de dia
para dia,
                                               até a última sílaba da recordação.
                                                 - Macbeth, V, v */
}



void receiver1(void)
{
  frame r;
  event_type event;                         /* preenchido por intervalo de espera, mas
não usado aqui */
```

```
  while (true) {
       wait_for_event(&event);            /* a única possibilidade é frame_arrival */
       from_physical_layer(&r);           /* obtém o quadro recebido */
       to_network_layer(&r.info);  /* repassa os dados à camada de rede */
  }
} [TN]
```

[F]Figura 3.10

[FL] Um protocolo simplex sem restrições

O receptor é igualmente simples. No início, ele espera que algo aconteça, e a única possibilidade é a chegada de um quadro não danificado. Eventualmente, o quadro chega e o procedimento *wait_for_event* retorna, com *event* definido como *frame_arrival* (o que, de qualquer forma, é ignorado). A chamada a *from_physical_layer* remove o quadro recém-chegado do buffer de hardware e o coloca na variável *r*, onde o código receptor poderá buscá-lo quando necessário. Por fim, a parte referente aos dados é repassada à camada de rede, e a camada de enlace de dados volta a esperar pelo próximo quadro, ficando efetivamente em suspenso até a chegada do quadro.

[T3] 3.3.2 Um protocolo simplex stop-and-wait

Agora, deixaremos de lado a restrição pouco realista utilizada no protocolo 1: a possibilidade de a camada de rede receptora processar os dados recebidos de uma forma infinitamente rápida (ou, o que é equivalente, a presença na camada de enlace de dados receptora de um espaço de buffer infinito, no qual poderão ser armazenados todos os quadros recebidos enquanto eles aguardam para serem processados). Continuamos supondo que o canal de comunicação não apresenta erros e que o tráfego de dados ainda é do tipo simplex.

O principal problema com que temos de lidar nesse caso é a forma de impedir que o transmissor inunde o receptor com dados, mais rapidamente do que este é capaz de processá-los. Em essência, se o receptor necessitar de um tempo Δt para executar *from_physical_layer* e *to_network_layer*, o transmissor terá de enviar os dados em uma velocidade média menor que um quadro por tempo Δt. Além disso, se considerarmos que não há nenhuma atividade automática de bufferização e enfileiramento no hardware do receptor, o transmissor nunca terá de enviar um novo quadro enquanto o mais antigo não tiver sido buscado por *from_physical_layer*, a menos que o novo quadro substitua o antigo.

Em determinadas circunstâncias restritas (por exemplo, transmissão síncrona e uma camada de enlace de dados receptora totalmente dedicada ao processamento da única linha de entrada), talvez seja possível para o transmissor simplesmente inserir um retardo no protocolo 1, a fim de reduzir sua velocidade e impedi-lo de sobrecarregar o receptor. No entanto, o mais comum é que cada camada de enlace de dados tenha várias linhas para processar, e que o intervalo de tempo entre a chegada de um quadro e seu processamento varie de forma considerável. Se puderem calcular o comportamento do receptor em uma situação totalmente desfavorável, os projetistas da rede serão capazes de programar o transmissor para funcionar tão lentamente que, mesmo quando todos os quadros sofrerem um retardo máximo, não haverá sobrecargas. O problema com essa estratégia é que ela é muito conservadora e nos leva a uma utilização da largura de banda muito abaixo do valor considerado ótimo, a menos que o comportamento do transmissor no melhor e no pior caso seja quase o mesmo (isto é, que a variação no tempo de reação da camada de enlace de dados seja muito pequena).

Uma solução mais geral para esse dilema é fazer o receptor enviar um feedback ao transmissor. Depois de enviar um pacote à sua camada de rede, o receptor

envia um pequeno quadro fictício (dummy) de volta ao transmissor, permitindo a transmissão do próximo quadro. Após o envio de um quadro, o protocolo exige que o transmissor espere sua vez, até a chegada do pequeno quadro fictício (isto é, da confirmação). A utilização de feedback do receptor para informar ao transmissor quando ele pode enviar mais dados é um exemplo do controle de fluxo mencionado anteriormente.

Os protocolos nos quais o transmissor envia um quadro e em seguida espera por uma confirmação antes de continuar sua operação são chamados **stop-and-wait**. A Figura 3.11 mostra um exemplo de protocolo simplex stop-and-wait.

Apesar de o tráfego de dados nesse exemplo ser simplex, indo apenas do transmissor ao receptor, há quadros sendo enviados em ambas as direções. Conseqüentemente, o canal de comunicação entre as duas camadas de enlace de dados deve ser capaz de realizar a transferência bidirecional de informações. No entanto, esse protocolo acarreta uma rígida alternância de fluxo: primeiro o transmissor envia um quadro, depois o receptor envia outro; em seguida, o transmissor envia mais um quadro e assim por diante. Um canal físico halfduplex seria suficiente nesse caso.

[arte: ver original da p. 207]

[TD]

/* O protocolo 2 (stop-and-wait) também implementa um fluxo de dados unidirecional

entre o transmissor e o receptor. Presume-se mais uma vez que o canal de comunicação

seja totalmente livre de erros, como no protocolo 1. No entanto, dessa vez, o receptor

tem buffer finito e uma velocidade de processamento finita; portanto, o protocolo

```
   deverá impedir explicitamente que o transmissor sobrecarregue o receptor
enviando
   dados mais rapidamente do que ele é capaz de processar. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                              /* buffer para um quadro enviado */
  packet buffer;                        /* buffer para um pacote enviado */
  event_type event;                     /* frame_arrival é a única possibilidade */

  while (true) {
        from_network_layer(&buffer);    /* obtém algo para enviar */
        s.info = buffer;                /* copia em s para transmissão */
        to_physical_layer(&s);          /* adeus, quadrinho */
        wait_for_event(&event);         /* não prossegue enquanto não recebe
permissão para ir em frente */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers para quadros */
  event_type event;                     /* frame_arrival é a única possibilidade */
  while (true) {
```

```
      wait_for_event(&event);           /* a única possibilidade é frame_arrival */
      from_physical_layer(&r);          /* obtém o quadro recebido */
      to_network_layer(&r.info);  /* repassa os dados à camada de rede */
      to_physical_layer(&s);            /* envia um quadro fictício para despertar
o transmissor */
  }
} [TN]
```

[F]Figura 3.11

[FL] Um protocolo simplex stop-and-wait

A exemplo do protocolo 1, o transmissor começa extraindo um pacote da camada de rede, utilizando-o para criar um quadro que em seguida é transmitido ao destino. Porém, agora, ao contrário do que ocorre no protocolo 1, o transmissor deve aguardar a chegada de um quadro de confirmação antes tornar a entrar em loop e buscar o próximo pacote da camada de rede. A camada de enlace de dados do transmissor não precisa sequer inspecionar o quadro recebido, pois só há uma possibilidade: o quadro recebido é sempre uma confirmação.

A única diferença entre *receptor1*e *receptor2* é que, após entregar um pacote à camada de rede, o *receptor2* envia um quadro de confirmação de volta ao transmissor, antes de entrar mais uma vez no loop de espera. Como apenas a chegada do quadro de volta ao transmissor é importante, e não seu conteúdo, o receptor não precisa incluir qualquer informação especifica no quadro.

[T3] 3.3.3 Um protocolo simplex para um canal com ruído

Agora, vamos considerar a situação normal de um canal de comunicação no qual ocorrem erros. Os quadros podem ser danificados ou completamente perdidos. No entanto, supomos que, se um quadro for danificado em trânsito, o hardware

receptor detectará essa ocorrência ao calcular o total de verificação. Se o quadro for danificado de tal forma que o total de verificação nunca esteja correto, uma possibilidade muito improvável, o protocolo em questão (e todos os outros protocolos) poderá apresentar falhas (isto é, poderá entregar um pacote incorreto à camada de rede).

À primeira vista, pode parecer que uma variação do protocolo 2 seria viável: a inclusão de um timer. O transmissor poderia enviar um quadro, mas o receptor só enviaria um quadro de confirmação se os dados fossem recebidos corretamente. Se um quadro danificado chegasse ao receptor, ele seria descartado. Após um certo tempo, o transmissor alcançaria seu timeout e enviaria o quadro mais uma vez. Esse processo seria repetido até que o quadro finalmente chegasse intacto. Esse esquema tem uma falha fatal. Pense no problema e tente descobrir o que poderia estar errado antes de continuar a leitura.

Para verificar o que poderia estar errado, lembre-se de que a função dos processos da camada de enlace de dados é oferecer comunicações transparentes e livres de erros entre os processos da camada de rede. A camada de rede da máquina *A* envia uma série de pacotes à camada de enlace de dados da mesma máquina. Esta, por sua vez, deve se certificar de que a camada de enlace de dados da máquina *B* enviará uma série idêntica de pacotes à camada de rede da mesma máquina. Em, particular, a camada de rede da máquina *B* não tem como saber se um pacote foi perdido ou duplicado; portanto, a camada de enlace de dados deve garantir que nenhuma combinação de erros de transmissão, mesmo improvável, possa fazer com que um pacote duplicado seja entregue a uma camada de rede.

Considere a seguinte situação:

1. A camada de rede de *A* envia o pacote 1 à sua camada de enlace de dados. O pacote é corretamente recebido em *B* e repassado à camada de rede de *B*. *B* envia

um quadro de confirmação de volta a *A*.

2. O quadro de confirmação se perde por completo. Ele simplesmente nunca chega ao destino. Tudo seria muito mais simples se o canal tivesse adulterado e perdido apenas quadros de dados, e não quadros de controle. No entanto, para nossa tristeza, o canal não faz distinção entre quadros.

3. Eventualmente, a camada de enlace de dados de *A* tem seu limite de tempo esgotado. Como não recebeu uma confirmação, ela presume (incorretamente) que seu quadro de dados se perdeu ou foi danificado e envia mais uma vez o quadro contendo o pacote 1.

4. O quadro duplicado também chega perfeitamente à camada de enlace de dados de *B* e é repassado de imediato, sem maiores problemas, à camada de rede. Caso *A* esteja enviando um arquivo a *B*, uma parte do arquivo será duplicada (isto é, a cópia do arquivo criado por *B* estará incorreta e o erro não será detectado). Em outras palavras, o protocolo falhará.

Na verdade, precisamos dar ao receptor alguma forma de poder distinguir entre um quadro que ele está recebendo pela primeira vez e uma retransmissão. A maneira mais fácil de conseguir isso é fazer o transmissor incluir um número de seqüência no cabeçalho de cada quadro enviado. Dessa forma, o receptor poderá verificar o número de seqüência de cada quadro recebido para confirmar se esse é um novo quadro ou se é uma duplicata a ser descartada.

Como aconselhamos a utilização de cabeçalhos não muito longos nos quadros, surge a seguinte pergunta: qual é a quantidade mínima de bits necessários para o número de seqüência? A única ambigüidade nesse protocolo ocorre entre um quadro m e seu sucessor direto, $m + 1$. Se o quadro m tiver sido perdido ou danificado, o receptor não o confirmará; portanto, o transmissor continuará tentando enviá-lo. Uma vez que o quadro tenha sido corretamente recebido, o receptor enviará uma confirmação de volta ao transmissor. É aqui que surge o

problema potencial. Dependendo do fato de o quadro de confirmação voltar ao transmissor corretamente ou não, o transmissor poderá tentar enviar m ou m + 1.

O evento que aciona o envio de m + 2 por parte do transmissor é a chegada de uma confirmação referente ao quadro m + 1. Porém, isso implica que m foi corretamente recebido pelo receptor e o mesmo aconteceu com sua confirmação em relação ao transmissor (caso contrário, o transmissor não teria começado a enviar m + 1 e muito menos m + 2). Conseqüentemente, nesse caso há uma única ambigüidade, presente entre um quadro e seu predecessor ou sucessor imediato, e não entre o predecessor e o sucessor propriamente ditos.

Um número de seqüência de 1 bit (0 ou 1) é, portanto, suficiente. A cada instante, o receptor espera o próximo número de seqüência. Qualquer quadro recebido que contenha o número de seqüência errado será rejeitado por ser considerado uma cópia. Quando um quadro contendo um número de seqüência correto chega, ele é aceito e repassado à camada de rede. Em seguida, o número de seqüência esperado é incrementado na base 2 (ou seja, 0 passa a ser 1 e 1 passa a ser zero).

Um exemplo desse tipo de protocolo é mostrado na Figura 3.12. Os protocolos nos quais o transmissor espera por uma confirmação positiva antes de passar para o próximo item de dados freqüentemente são chamados **PAR (Positive Acknowledgement with Retransmission — confirmação positiva com retransmissão)** ou **ARQ (Automatic Repeat reQuest — solicitação de repetição automática)**. A exemplo do protocolo 2, esse protocolo também transmite dados em apenas um sentido.

O protocolo 3 difere de seus predecessores pelo fato de tanto o transmissor quanto o receptor terem uma variável cujo valor é memorizado enquanto a camada de enlace de dados se encontra em estado de espera. Em

next_frame_to_send, o transmissor armazena o número de seqüência do próximo quadro a ser enviado, e em *frame_expected* o receptor armazena o número de seqüência do próximo quadro esperado. Cada protocolo tem uma breve fase de inicialização antes de entrar no loop infinito.

[arte: ver original da p. 210]

[TD]

```
/* O protocolo 3 (par) permite um fluxo de dados unidirecional por um canal não
confiável. */
#define MAX_SEQ 1                                   /* deve ser 1 para o protocolo
3 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void sender3(void)
{
  seq_nr next_frame_to_send;                        /* número de seqüência do */
                                                    /* próximo quadro a ser
enviado */
  frame s;                                          /* variável de rascunho */
  packet buffer;                                    /* buffer para um pacote
enviado */
  event_type event;
  next_frame_to_send = 0;                           /* inicializa os números de
seqüência */
                                                    /* dos quadros a serem
enviados */
  from_network_layer(&buffer);                      /* busca primeiro pacote */
  while (true) {
```

```
      s.info = buffer;                        /* constrói um quadro para
transmissão */
      s.seq = next_frame_to_send;             /* insere número de
seqüência em quadro */
      to_physical_layer(&s);                  /* envia quadro ao destino */
      start_timer(s.seq);                     /* se a resposta demorar
demais, ativa timeout */
      wait_for_event(&event);                 /* frame_arrival, cksum_err,
timeout */
      if (event == frame_arrival) {
            from_physical_layer(&s);          /* recebe a confirmação */
            if (s.ack == next_frame_to_send) {
                  stop_timer(s.ack);          /* desativa o timer */
                  from_network_layer(&buffer);  /* obtém o próximo quadro a
enviar */
                  inc(next_frame_to_send);  /* inverte next_frame_to_send */
            }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;
  frame_expected = 0;
```

```
  while (true) {
       wait_for_event(&event);                    /* possibilidades:
frame_arrival, cksum_err */
       if (event == frame_arrival) {              /* chegou um quadro válido.
*/
            from_physical_layer(&r);              /* recebe o quadro recém-
chegado */
            if (r.seq == frame_expected) {        /* esse é o quadro que
estávamos esperando. */
                 to_network_layer(&r.info);       /* repassa os dados à camada
de rede */
                 inc(frame_expected);             /* na próxima vez, espera o
outro número de seqüência */
            }
            s.ack = 1 - frame_expected;           /* informa qual quadro está
sendo confirmado */
            to_physical_layer(&s);           /* envia confirmação */
       }
  }
} [TN]
```

[F]Figura 3.12

[FL] Uma confirmação positiva com protocolo de retransmissão

Após enviar um quadro, o transmissor ativa o timer. Caso já esteja ativado, o timer será reinicializado para permitir a contagem de outro intervalo. O intervalo deve ser definido de forma que haja tempo suficiente para o quadro chegar ao receptor e ser processado e para o quadro de confirmação ser enviado de volta ao

transmissor. Somente quando o intervalo de tempo tiver se esgotado, poderemos supor com segurança que o quadro transmitido ou sua confirmação se perdeu, e que será necessário enviar uma cópia. Se o intervalo de timeout for definido com um valor curto demais, o transmissor irá enviar quadros desnecessários. Embora não afetem a correção do protocolo, esses quadros extras prejudicarão o desempenho.

Depois de transmitir um quadro e ativar o timer, o transmissor espera que algo interessante aconteça. Existem apenas três possibilidades: o quadro de confirmação chegar sem danos, o quadro de confirmação chegar com erro ou o timer ser desativado. Se uma confirmação válida for recebida, o transmissor buscará o próximo pacote em sua camada de rede e o colocará no buffer, substituindo o pacote anterior. Ele também aumentará o número de seqüência. Se for recebido um quadro com erro ou se não chegar nenhum quadro ao destino, o buffer e o número de seqüência permanecerão inalterados; nesse caso, uma cópia do quadro poderá ser enviada.

Quando um quadro válido chega ao receptor, seu número de seqüência é conferido, para verificar se ele é uma cópia. Se não for uma cópia, o quadro será aceito, enviado à camada de rede, e uma confirmação será gerada. Cópias e quadros danificados não serão repassados à camada de rede.

[T2] 3.4 Protocolos de janela deslizante

Nos protocolos apresentados anteriormente, os quadros de dados eram transmitidos em apenas um sentido. Em situações mais práticas, há necessidade de transmitir dados em ambos os sentidos. Você pode obter uma transmissão de dados full-duplex definindo dois canais de comunicação distintos e usar cada um deles para um tráfego de dados simplex (em diferentes sentidos). Se isso for feito, haverá dois circuitos físicos separados, cada um com um canal "direto"

(para dados) e um canal "inverso" (para confirmações). Em ambos os casos, a largura de banda do canal inverso é quase totalmente perdida. Na verdade, o usuário está pagando por dois circuitos, mas está usando apenas a capacidade de um deles.

Uma idéia melhor é usar o mesmo circuito para dados em ambos os sentidos. Afinal de contas, nos protocolos 2 e 3 ele já estava sendo usado para transmitir quadros em ambos os sentidos, e o canal inverso tem a mesma capacidade do canal direto. Nesse modelo, os quadros de dados enviados de *A* para *B* são misturados com os quadros de confirmação enviados de *A* para *B*. Ao verificar o campo *kind* do cabeçalho de um quadro recebido, o receptor pode identificar se o quadro é de dados ou de confirmação.

Apesar de o entrelaçamento de quadros de dados e de controle no mesmo circuito representar um avanço em relação ao uso de dois circuitos físicos separados, ainda é possível introduzir mais um aperfeiçoamento. Quando um quadro de dados chega a seu destino, em vez de enviar imediatamente um quadro de controle separado, o receptor se contém e espera até a camada de rede enviar o próximo quadro. A confirmação é acrescentada ao quadro de dados que está sendo enviado (por meio do campo *ack* do cabeçalho de quadro). Na verdade, a confirmação pega carona no próximo quadro de dados que estiver sendo enviado. A técnica de retardar temporariamente as confirmações e enviá-las junto com o próximo quadro de dados é conhecida pelo nome de **piggybacking** (superposição).

A principal vantagem do piggybacking em relação ao envio de quadros de confirmação distintos é a melhor utilização da largura de banda disponível para o canal. O campo *ack* do cabeçalho de quadro precisa de apenas alguns bits, enquanto um quadro separado precisaria de um cabeçalho, da confirmação e de um total de verificação. Além disso, um número menor de quadros enviados

significa menor quantidade de interrupções de "chegada de quadro", e talvez menor quantidade de buffers no receptor, dependendo da forma como o software do receptor está organizado. No próximo protocolo a ser examinado, o campo de piggyback necessita apenas de um bit no cabeçalho de quadro. Em geral, ele raramente precisa de mais que alguns bits no cabeçalho.

No entanto, o piggybacking introduz uma complicação não presente em confirmações separadas. Quanto tempo a camada de enlace de dados deve esperar por um pacote ao qual deverá acrescentar a confirmação? Se a camada de enlace de dados esperar durante um intervalo de tempo maior que o permitido pelo timeout do transmissor, o quadro será retransmitido, o que invalidará todo o processo de confirmação. Se a camada de enlace de dados fosse um oráculo e pudesse prever o futuro, ela saberia quando o próximo pacote da camada de rede estivesse chegando e poderia decidir entre esperar por ele ou enviar imediatamente uma confirmação separada, dependendo da duração prevista do tempo de espera. É óbvio que a camada de enlace de dados não é capaz de prever o futuro; portanto, ela deve recorrer a algum esquema *ad hoc*, como esperar durante um número fixo de milissegundos. Se um novo pacote chegar logo, a confirmação será acrescentada a ele; caso contrário, se nenhum pacote tiver chegado até o final desse intervalo de tempo, a camada de enlace de dados simplesmente enviará um quadro de confirmação separado.

Os três protocolos seguintes são protocolos bidirecionais que pertencem a uma classe de protocolos identificados como protocolos de **janela deslizante**. Os três apresentam diferenças em termos de eficiência, complexidade e requisitos de buffer, como discutiremos mais adiante. Nesses protocolos, como em todos os protocolos de janela deslizante, cada quadro enviado contém um número de seqüência, variando desde 0 até algum valor máximo. Em geral, o valor máximo e $2^n - 1$, de forma que o número de seqüência caiba exatamente em um campo de

n bits. O protocolo de janela deslizante stop-and-wait utiliza $n = 1$, restringindo os números de seqüência a 0 e 1; no entanto, versões mais sofisticadas podem usar um valor arbitrário de *n*.

A essência de todos os protocolos de janela deslizante é o fato de que, em qualquer instante, o transmissor mantém um conjunto de números de seqüência correspondentes a quadros que ele pode enviar. Dizemos que esses quadros estão reunidos na **janela de transmissão.** Da mesma forma, o receptor mantém uma **janela de recepção** correspondente ao conjunto de quadros que está apto a aceitar. A janela do transmissor e a janela do receptor não precisam ter os mesmos limites superior e inferior ou o mesmo tamanho. Em alguns protocolos, essas janelas têm tamanho fixo, mas em outros elas podem aumentar e diminuir à medida que os quadros são enviados e recebidos.

Apesar desses protocolos permitirem que a camada de enlace de dados tenha mais liberdade em relação à ordem em que poderá enviar e receber quadros, definitivamente não descartamos o requisito de que o protocolo deve entregar os pacotes à camada de rede na mesma ordem em que eles foram repassados à camada de enlace de dados da máquina transmissora. Outra exigência que não mudou é que o canal de comunicação física entregue todos os quadros na ordem em que eles são enviados.

Os números de seqüência contidos na janela do transmissor representam quadros que foram enviados ou que podem ser enviados, mas ainda não confirmados. Sempre que chega um novo pacote da camada de rede, ele recebe o próximo número de seqüência mais alto, e a borda superior da janela é incrementada em uma unidade. Quando uma confirmação é recebida, a borda inferior é incrementada em uma unidade. Dessa forma, a janela mantém continuamente uma lista de quadros não confirmados. A Figura 3.13 mostra um exemplo.

[arte: ver original p. 213]

[Dísticos]

[1]Transmissor

[2]Receptor

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 3.13

[FL]Uma janela deslizante de tamanho 1, com um número de seqüência de 3 bits. (a) Inicialmente. (b) Depois que o primeiro quadro é enviado. (c) Depois que o primeiro quadro é recebido. (d) Depois que a primeira confirmação é recebida

Tendo em vista que os quadros atualmente presentes na janela do transmissor podem ser perdidos ou danificados em trânsito, o transmissor deve manter todos esses quadros em sua memória para que a retransmissão seja possível. Assim, se o tamanho máximo da janela for *n*, o transmissor precisará de *n* buffers para armazenar os quadros não confirmados. Se a janela chegar a seu tamanho máximo, a camada de enlace de dados do transmissor será obrigada a desativar a camada de rede até que outro buffer esteja livre.

O tamanho da janela da camada de enlace de dados receptora corresponde aos quadros que ela é capaz de aceitar. Qualquer quadro que ficar fora da janela será simplesmente descartado. Quando for recebido um quadro cujo número de seqüência é igual à borda inferior da janela, ele será repassado à camada de rede, será gerada uma confirmação, e a janela será incrementada em uma unidade. Ao contrário da janela do transmissor, a janela do receptor sempre permanece com seu tamanho inicial. Observe que um tamanho de janela igual a 1 significa que a camada de enlace de dados só aceita quadros em ordem, mas para janelas maiores isso não é verdade. Em contraste, a camada de rede sempre recebe

dados na ordem adequada, independente do tamanho da janela da camada de enlace de dados.

A Figura 3.13 mostra um exemplo com um tamanho máximo de janela igual a 1. Inicialmente, não há quadros pendentes; portanto, as bordas inferior e superior da janela do transmissor são iguais mas, à medida que o tempo passa, a situação se desenvolve da maneira mostrada.

[T3] 3.4.1 Um protocolo de janela deslizante de um bit

Antes de abordarmos o caso geral, vamos examinar primeiro um protocolo de janela deslizante com um tamanho máximo de janela igual a 1. Esse tipo de protocolo utiliza o stop-and-wait, pois o transmissor envia um quadro e aguarda sua confirmação antes de enviar o quadro seguinte.

A Figura 3.14 representa esse tipo de protocolo. Assim como os demais, esse protocolo começa definindo algumas variáveis. *Next_frame_to_send* informa qual quadro o transmissor está tentando enviar. De modo semelhante, *frame_expected* informa que quadro o receptor está esperando. Nos dois casos, 0 e 1 são as únicas possibilidades.

Normalmente, uma das duas camadas de enlace de dados parte primeiro transmite o primeiro quadro. Em outras palavras, apenas um dos programas da camada de enlace de dados deve conter as chamadas de procedimento *to_physical_layer* e *start_timer* fora do loop principal. No caso de as duas camadas de enlace de dados partirem simultaneamente, surgirá uma situação peculiar, que será discutida mais adiante. A máquina que inicia busca o primeiro pacote em sua camada de rede, constrói um quadro a partir dele e o envia. Quando esse (ou qualquer) quadro chega ao destino, a camada de enlace de dados receptora verifica se ele é uma cópia, como ocorreu no protocolo 3. Se o quadro for o esperado, ele será repassado à camada de rede e a janela do

receptor será deslocada para cima.

O campo de confirmação contém o número do último quadro recebido sem erro. Se esse número estiver de acordo com o número de seqüência do quadro que o transmissor está tentando enviar, o transmissor saberá que já cuidou do quadro armazenado em *buffer* e poderá buscar o pacote seguinte em sua camada de rede. Se o número de seqüência for discordante, o transmissor deve continuar tentando enviar o mesmo quadro. Sempre que um quadro é recebido, um outro quadro também é enviado de volta.

Agora, vamos examinar o protocolo 4 para ver o quanto ele é flexível em relação a situações patológicas. Suponha que o computador *A* esteja tentando enviar seu quadro 0 ao computador *B* e que *B* esteja tentando enviar seu quadro 0 ao computador *A*. Imagine que *A* envia um quadro a *B*, mas o intervalo de timeout de A é curto demais. Conseqüentemente, A pode completar o timeout repetidas vezes, enviando uma série de quadros idênticos, todos com *seq* = 0 e *ack* = 1. Quando o primeiro quadro válido chegar a *B*, ele será aceito, e *frame_expected* será definido como 1. Todos os quadros subseqüentes serão rejeitados, porque *B* agora está esperando quadros com número de seqüência 1, e não 0. Além disso, como todas as cópias têm *ack* = 1 e *B* ainda está aguardando uma confirmação de 0, *B* não buscará um novo pacote em sua camada de rede.

[arte: ver original da p. 215]

[TD]

```
/* O protocolo 4 (de janela deslizante) é bidirecional. */
#define MAX_SEQ 1                                    /* deve ser 1 para o protocolo
4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
```

```
{
  seq_nr next_frame_to_send;                    /* 0 ou 1 somente */
  seq_nr frame_expected;                   /* 0 ou 1 somente */
  frame r, s;                                   /* variáveis de rascunho */
  packet buffer;                                /* pacote que está sendo
enviado no momento */
  event_type event;
  next_frame_to_send = 0;                       /* próximo quadro no fluxo
de saída */
  frame_expected = 0;                           /* quadro esperado em
seguida */
  from_network_layer(&buffer);                  /* busca um pacote na
camada de rede */
  s.info = buffer;                              /* prepara-se para enviar o
quadro inicial */
  s.seq = next_frame_to_send;                   /* insere número de
seqüência no quadro */
  s.ack = 1 - frame_expected;                   /* confirmação com piggyback
*/
  to_physical_layer(&s);                        /* transmite o quadro */
  start_timer(s.seq);                           /* inicializa o timer atual */
  while (true) {
       wait_for_event(&event);                  /* frame_arrival, cksum_err ou
timeout */
       if (event == frame_arrival) {            /* um quadro chegou sem
danos. */
              from_physical_layer(&r);          /* vai buscá-lo */
```

```
        if (r.seq == frame_expected) {          /* trata o fluxo de quadros
recebidos. */
            to_network_layer(&r.info);          /* repassa pacote à camada
de rede */
            inc(frame_expected);                /* inverte próximo número de
seqüência esperado */
        }
        if (r.ack == next_frame_to_send) {      /* trata o fluxo de quadros
enviados. */
            stop_timer(r.ack);                  /* desativa o timer */
            from_network_layer(&buffer);        /* busca novo pacote na
camada de rede */
            inc(next_frame_to_send);            /* inverte número de
seqüência do transmissor */
        }
    }
    s.info = buffer;                            /* constrói quadro a ser
enviado */
    s.seq = next_frame_to_send;                 /* insere número de
seqüência no quadro */
    s.ack = 1 – frame_expected;                 /* número de seqüência do
último quadro recebido */
    to_physical_layer(&s);                      /* transmite um quadro */
    start_timer(s.seq);                         /* inicializa o timer atual */
  }
} [TN]
```

[F]Figura 3.14

[FL] Um protocolo de janela deslizante de um bit

Após a chegada de todas as cópias rejeitadas, *B* enviará um quadro para *A* contendo *seq* = 0 e *ack* = 0. Eventualmente, um desses quadros chegará sem erros à máquina *A*, fazendo com que *A* comece a enviar o próximo pacote. Nenhuma combinação de quadros perdidos ou timeouts prematuros pode fazer o protocolo entregar pacotes duplicados à camada de rede, ignorar um pacote ou chegar a um impasse.

Entretanto, surgirá uma situação peculiar se os dois lados enviarem simultaneamente um pacote inicial. Essa dificuldade de sincronização está ilustrada na Figura 3.15. Na parte (a), é exibida a operação normal do protocolo. Na parte (b), observamos a peculiaridade. Se *B* esperar pelo primeiro quadro de *A* antes de enviar um de seus quadros, a seqüência será a da parte (a), e todos os quadros serão aceitos. Porém, se *A* e *B* iniciarem a comunicação ao mesmo tempo, seus primeiros quadros se cruzarão e as camadas de enlace de dados recairão na situação (b). Em (a), cada quadro recebido traz um novo pacote para a camada de rede; não há cópias. Em (b), metade dos quadros contém cópias, embora não haja erros de transmissão. Situações similares podem ocorrer como resultado de timeouts prematuros, mesmo quando está claro que um lado começa primeiro. Na verdade, se ocorrerem vários timeouts prematuros, os quadros poderão ser enviados três vezes ou mais.

[arte: ver original p. 216]

[Dísticos]

[1]A envia (0, 1, A0)

B recebe (0, 1, A0)*

B envia (0, 0, B0)

A recebe (0, 0, B0)*

A envia (1, 0, A1)

B recebe (1, 0, A1)

B envia (1, 1, B1)

A recebe (1, 1, B1)*

A envia (0, 1, A2)

B recebe (0, 1, A2)*

B envia (0, 0, B2)

A recebe (0, 0, B2)*

A envia (1, 0, A3)

B recebe (1, 0, A3)*

B envia (1, 1, B3)

[2]A envia (0, 1, A0)

B envia (0, 1, B0)

B recebe (0, 1, A0)*

B envia (0, 0, B0)

A recebe (0, 1, B0)*

A envia (0, 0, A0)

B recebe (0, 0, A0)

B envia (1, 0, B1)

A recebe (0, 0, B0)

A envia (1, 0, A1)

B recebe (1, 0, A1)*

B envia (1, 1, B1)

A recebe (1, 0, B1)*

A envia (1, 1, A1)

B recebe (1, 1, A1)

B envia (0, 1, B2)

[3] Tempo

(a) (b)

[F]Figura 3.15

[FL] Dois cenários referentes ao protocolo 4. (a) Caso normal. (b) Caso anormal. A notação é (seqüência, confirmação, número do pacote). Um asterisco indica onde uma camada de rede aceita um pacote

[T3] 3.4.2 Um protocolo que utiliza go back n

Até agora estávamos supondo implicitamente que o tempo de transmissão necessário para a chegada de um quadro até o receptor somado ao tempo de transmissão para o retorno da confirmação era insignificante. Às vezes, essa suposição é nitidamente falsa. Nessas situações, o longo tempo de viagem de ida e volta pode ter implicações importantes para a eficiência da utilização da largura de banda. Como exemplo, considere um canal de satélite de 50 kbps com um retardo de propagação de ida e volta de 500 ms. Vamos imaginar a tentativa de usar o protocolo 4 para enviar quadros de 1.000 bits pelo satélite. Em $t = 0$, o transmissor começa a enviar o primeiro quadro. Em $t = 20$ ms, o quadro já foi completamente enviado. Até $t = 270$ ms, o quadro ainda não chegou completamente ao receptor, e até $t = 520$ ms, na melhor das hipóteses, a confirmação ainda não voltou ao transmissor (sem nenhum tempo de espera no receptor e com um quadro de confirmação curto). Isso significa que o transmissor esteve bloqueado durante 500/520 ou 96% do tempo (isto é, apenas 4% da largura de banda disponível foram utilizados). É claro que a combinação de um longo tempo de trânsito, alta largura de banda e pequeno comprimento de quadro é desastrosa em termos de eficiência.

O problema descrito anteriormente pode ser visto como uma conseqüência da regra que exige que um transmissor espere por uma confirmação antes de enviar

outro quadro. Se essa restrição não for rigorosa, poderemos obter uma eficiência muito melhor. Basicamente, a solução está em permitir que o transmissor envie até w quadros antes do bloqueio, e não apenas 1. Com uma escolha apropriada de w, o transmissor será capaz de transmitir quadros continuamente durante um tempo igual ao tempo de trânsito da viagem de ida e volta, sem ocupar a janela toda. No exemplo anterior, w deve ser pelo menos igual a 26. O transmissor começa enviando o quadro 0 como antes. Decorrido o tempo para o término do envio de 26 quadros, em $t = 520$, a confirmação do quadro 0 terá acabado de chegar. Daí em diante, as confirmações chegarão a cada 20 ms, e assim o transmissor sempre terá permissão para continuar exatamente quando precisar dela. A todo momento ficam pendentes 25 ou 26 quadros não confirmados. Em outras palavras, o tamanho máximo da janela do transmissor é 26.

A necessidade de uma janela grande do lado transmissor surge sempre que o produto da largura de banda pelo retardo de ida e volta é grande. Se a largura de banda for alta, mesmo para um retardo moderado, o transmissor esgotará sua janela rapidamente, a menos que ele tenha uma janela grande. Se o retardo for alto (por exemplo, em um canal de satélite geoestacionário), o transmissor irá esgotar sua janela até mesmo no caso de uma largura de banda moderada. O produto desses dois fatores informa basicamente qual é a capacidade do canal, e o transmissor precisa ser capaz de preenchê-lo sem interrupções, a fim de operar com eficiência máxima.

Essa técnica é conhecida como **pipelining**. Se a capacidade do canal for b bits/s, se o tamanho do quadro for l bits e o tempo de propagação da viagem de ida e volta for R segundos, o tempo necessário para a transmissão de um único quadro será l/b segundos. Depois que o último bit de um quadro de dados tiver sido enviado, haverá um retardo $R/2$ antes desse bit chegar ao receptor, e outro retardo de pelo menos $R/2$ até o recebimento da confirmação, totalizando um

retardo igual a R. No algoritmo de protocolo stop-and-wait, a linha está ocupada durante o tempo l/b e ociosa durante o tempo R, o que resulta em

utilização da linha = $l/(l + bR)$

Se $l < bR$, a eficiência será inferior a 50%. Tendo em vista que sempre existe um retardo diferente de zero para a propagação de retorno da confirmação, em princípio o pipelining pode ser usado para manter a linha ocupada durante esse intervalo. No entanto, se o intervalo for pequeno, a complexidade adicional não valerá a pena.

O pipelining de quadros em um canal de comunicação não confiável faz surgir algumas questões muito sérias. Primeiro, o que acontecerá se um quadro em meio a um longo fluxo for danificado ou perdido? Um grande número de quadros sucessivos chegará ao receptor antes mesmo que o transmissor descubra que algo está errado. Quando um quadro danificado chega ao receptor, ele deve sem dúvida ser descartado. No entanto, o que o receptor deve fazer com todos os quadros corretos que o seguem? Lembre-se de que a camada de enlace de dados receptora é obrigada a entregar pacotes à camada de rede em seqüência. Na Figura 3.16, observamos os efeitos do pipelining sobre a recuperação de erros. Agora, vamos examiná-lo em detalhes.

[arte: ver original p. 218]

[Dísticos]

[1]Intervalo de timeout

[2]Erro Quadros descartados pela camada de enlace de dados

Tempo

(a)

[3]Erro Quadros inseridos no buffer pela camada de enlace de dados

(b)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 3.16

[FL] Pipelining e recuperação de erros. Efeito de um erro quando (a) o tamanho da janela receptora é igual a 1 e quando (b) o tamanho da janela receptora é grande

Há duas estratégias básicas para lidar com erros na presença do pipelining. De acordo com uma delas, denominada **go back n**, o receptor simplesmente descarta todos os quadros subseqüentes e não envia qualquer confirmação desses quadros descartados. Essa estratégia corresponde a uma janela de recepção de tamanho 1. Em outras palavras, a camada de enlace de dados se recusa a aceitar qualquer quadro, exceto o próximo quadro que ela tem de entregar à camada de rede. Se a janela do transmissor for totalmente preenchida antes do timer encerrar a contagem, o pipeline começará a se esvaziar. Conseqüentemente, o transmissor interromperá a transmissão e retransmitirá todos os quadros não confirmados em ordem, começando pelo quadro danificado ou perdido. Essa abordagem poderá desperdiçar uma grande quantidade de largura de banda, se a taxa de erros for alta.

Na Figura 3.16(a), vemos go back n para o caso em que a janela do receptor é grande. Os quadros 0 e 1 são corretamente recebidos e confirmados. Porém, o quadro 2 está danificado ou perdido. O transmissor, desavisado desse problema, continua a enviar quadros até expirar o timer correspondente ao quadro 2. Em seguida, ele volta até o quadro 2 e começa tudo de novo a partir dele, enviando mais uma vez os quadros 2, 3, 4 etc.

A outra estratégia geral para tratamento de erros quando é feito o pipelining de quadros denomina-se **retransmissão seletiva** (selective repeat). Quando ela é utilizada, um quadro incorreto recebido é descartado, mas os quadros sem

defeitos recebidos depois dele são inseridos no buffer. Quando o transmissor chega ao timeout, apenas o quadro não confirmado mais antigo é retransmitido. Se esse quadro chegar corretamente, o receptor poderá entregar à camada de rede, em seqüência, todos os quadros que armazenou no buffer. Com freqüência, a retransmissão seletiva é combinada com a ação de fazer o receptor enviar uma confirmação negativa (NAK – negative acknowledgement) ao detectar um erro, por exemplo, quando receber um erro de total de verificação ou um quadro fora de seqüência. As NAKs estimulam a retransmissão antes de expirar o timer correspondente e, desse modo, melhoram o desempenho.

Na Figura 3.16(b), os quadros 0 e 1 são mais vez recebidos e confirmados corretamente, e o quadro 2 é perdido. Quando o quadro 3 chega ao receptor, a camada de enlace de dados do receptor percebe que perdeu um quadro, e assim envia de volta uma NAK correspondente ao quadro 2, mas armazena no buffer o quadro 3. Quando os quadros 4 e 5 chegam, eles também são inseridos no buffer pela camada de enlace de dados, em vez de serem repassados à camada de rede. Eventualmente, a NAK do quadro 2 volta ao transmissor, que retransmite de imediato o quadro 2. Quando esse quadro chega, a camada de enlace de dados fica com os quadros 2, 3, 4 e 5, e pode repassar todos eles à camada de rede na ordem correta. Ela também pode confirmar todos os quadros até o quadro 5, inclusive, como mostra a figura. Se a NAK se perder, o transmissor chegará ao timeout correspondente ao quadro 2 e o enviará (e apenas esse quadro) por sua própria iniciativa, mas isso pode acontecer um pouco mais tarde. Na realidade, a NAK acelera a retransmissão de um quadro específico.

A estratégia de retransmissão seletiva corresponde a uma janela receptora maior que 1. Qualquer quadro que estiver dentro da janela pode ser aceito e colocado no buffer até todos os quadros precedentes terem sido transmitidos à camada de rede. Essa abordagem poderá exigir um volume de memória muito grande da

camada de enlace de dados, caso a janela seja muito grande.

Esses dois enfoques alternativos traduzem compromissos entre largura de banda e espaço no buffer da camada de enlace de dados. Dependendo de qual recurso seja mais escasso, um ou outro poderá ser usado. A Figura 3.17 mostra um protocolo de pipelining no qual a camada de enlace de dados receptora aceita apenas quadros em ordem; os quadros que vierem depois de um quadro com erro serão descartados. Nesse protocolo, abandonamos pela primeira vez a suposição de que a camada de rede sempre tem um suprimento infinito de pacotes a enviar. Quando a camada de rede tem um pacote que deseja enviar, ela pode provocar a ocorrência de um evento *nevwork_layer_ready*. Entretanto, para reforçar a regra de controle de fluxo que não permite mais de *MAX_SEQ* quadros não confirmados pendentes em qualquer instante, a camada de enlace de dados deve ser capaz de proibir a camada de rede de sobrecarregá-la com mais trabalho. Os procedimentos de biblioteca *enable_network_layer* e *disable_network_layer* executam essa função.

[arte: ver original da p. 220]

[TD]

/* O protocolo 5 (go back n) permite a existência de muitos quadros pendentes. O transmissor poderá

transmitir até MAX_SEQ quadros sem a necessidade de esperar por uma confirmação.

Além disso, ao contrário dos protocolos anteriores, não presumimos que a camada de rede

está sempre recebendo um novo pacote. Em vez disso, a camada de rede provoca um

evento network_layer_ready quando há um pacote a ser enviado. */

```
#define MAX_SEQ 7                          /* deve ser 2^n - 1 */

typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready}

event_type;

#include "protocol.h"


static boolean between(seq_nr a, seq_nr b, seq_nr c)

{

/* Retorna true se a <= b < c, de forma circular; caso contrário, retorna false. */

  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))

        return(true);

    else

        return(false);

}


static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])

{

/* Constrói e envia um quadro de dados. */

  frame s;                                 /* variável de rascunho */


  s.info = buffer[frame_nr];                         /* insere pacote em quadro */

  s.seq = frame_nr;                        /* insere número de seqüência em quadro

*/

  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); /* confirmação com

piggyback */

  to_physical_layer(&s);                             /* transmite o quadro */

  start_timer(frame_nr);                             /* inicializa o timer atual */

}
```

```
void protocol5(void)
{
  seq_nr next_frame_to_send;              /* MAX_SEQ > 1; usado para fluxo
enviado */
  seq_nr ack_expected;                    /* quadro mais antigo ainda não
confirmado */
  seq_nr frame_expected;            /* próximo quadro esperado no fluxo
recebido */
  frame r;                                /* variável de rascunho */
  packet buffer[MAX_SEQ + 1];             /* buffers para o fluxo enviado */
  seq_nr nbuffered;                 /* # buffers de saída atualmente em uso */
  seq_nr i;                               /* usada para indexar no array do
buffer */
  event_type event;

  enable_network_layer();                 /* ativa eventos network_layer_ready
*/
  ack_expected = 0;                       /* próxima confirmação esperada
recebida */
  next_frame_to_send = 0;                 /* próximo quadro de saída */
  frame_expected = 0;                     /* número de quadro recebido
esperado */
  nbuffered = 0;                          /* inicialmente, nenhum pacote é
inserido no buffer */
  while (true) {
     wait_for_event(&event);              /* quatro possibilidades: veja
```

```
event_type anterior */

    switch(event) {
      case network_layer_ready:                /* a camada de rede tem um pacote
a enviar */
           /* Aceita, salva e transmite um novo quadro. */
           from_network_layer(&buffer[next_frame_to_send]); /* busca novo pacote
*/
           nbuffered = nbuffered + 1;          /* expande a janela do transmissor
*/
           send_data(next_frame_to_send, frame_expected, buffer);     /*
transmite o quadro */
           inc(next_frame_to_send);            /* avança a borda superior da janela
do transmissor */
           break;

      case frame_arrival:                      /* chegou um quadro de dados ou
de controle */
            from_physical_layer(&r);           /* busca quadro recebido na camada
física */

            if (r.seq == frame_expected) {
                  /* Quadros só são aceitos em ordem. */
                  to_network_layer(&r.info);   /* repassa pacote à camada de rede
*/
                  inc(frame_expected);         /* avança borda inferior da janela do
receptor */
```

```
                }

                /* Confirmação de n implica n – 1, n – 2 etc.  Verifique isso. */
                while (between(ack_expected, r.ack, next_frame_to_send)) {
                        /* Trata confirmação com piggyback. */
                        nbuffered = nbuffered – 1;   /* um quadro a menos no buffer */
                        stop_timer(ack_expected);    /* quadro chegou intacto;
interrompe timer */
                        inc(ack_expected);           /* contrai janela do transmissor */
                }
                break;

        case cksum_err: break;                       /* simplesmente ignora quadros
incorretos */

        case timeout:                                /* problema; retransmite todos os
quadros pendentes */
                next_frame_to_send = ack_expected; /* inicia retransmissão aqui */
                for (i = 1; i <= nbuffered; i++) {
                        send_data(next_frame_to_send, frame_expected, buffer);      /*
envia quadro novamente */
                        inc(next_frame_to_send);   /* prepara-se para enviar o próximo */
                }

    }

    if (nbuffered < MAX_SEQ)
```

```
            enable_network_layer();
     else
            disable_network_layer();
  }
} [TN]
```

[F]Figura 3.17

[FL] Um protocolo de janela deslizante que utiliza go back n

Observe que no máximo *MAX_SEQ* quadros, e não *MAX_SEQ* + 1, podem estar pendentes em qualquer instante, mesmo que haja *MAX_SEQ* + 1 números de seqüência distintos: 0, 1, 2, ..., *MAX_SEQ*. Para saber por que essa restrição é necessária, considere a situação a seguir, com *MAX_SEQ* =7.

1. O transmissor envia quadros de 0 a 7.
2. Uma confirmação com piggyback (de carona) para o quadro 7 volta eventualmente ao transmissor.
3. O transmissor envia mais oito quadros, novamente com números de seqüência de 0 a 7.
4. Agora chega outra confirmação com piggyback correspondente ao quadro 7.

A questão é: os oito quadros pertencentes ao segundo lote chegaram com sucesso, ou todos eles se perderam (a contagem descarta os quadros posteriores a um erro, considerando-os perdidos)? Nos dois casos, o receptor estaria enviando o quadro 7 como confirmação. O transmissor não tem como saber disso. Por essa razão, o número máximo de quadros pendentes deve se restringir a *MAX_SEQ*.

Apesar de não armazenar no buffer os quadros recebidos após um quadro com erro, o protocolo 5 não escapa totalmente ao problema do armazenamento em buffer. Tendo em vista que um transmissor talvez seja obrigado a retransmitir

todos os quadros não confirmados em um determinado momento no futuro, ele deverá reter todos os quadros transmitidos até ter certeza de que eles foram aceitos pelo receptor. Quando uma confirmação chega para o quadro *n*, os quadros *n* – 1, *n* – 2 e assim por diante também são confirmados de forma automática. Essa propriedade é especialmente importante nos casos em que alguns dos quadros anteriores que representavam confirmações se perderam ou foram adulterados. Sempre que uma confirmação chega, a camada de enlace de dados verifica se algum buffer pode ser liberado. Se os buffers puderem ser liberados (isto é, se houver espaço disponível na janela), uma camada de rede bloqueada anteriormente poderá ter permissão para provocar mais eventos *network_layer_ready*.

Para esse protocolo, supomos que sempre existe tráfego no sentido inverso, para que as confirmações possam ser transportadas por piggyback. Se não houver tráfego inverso, nenhuma confirmação poderá ser enviada. O protocolo 4 não precisa dessa suposição, pois ele envia um quadro de volta toda vez que recebe um quadro, mesmo que tenha acabado de enviar esse quadro. No próximo protocolo, resolveremos de modo elegante o problema do tráfego de mão única.

Por ter vários quadros pendentes, é claro que o protocolo 5 necessita de vários timers, um para cada quadro pendente. Cada quadro tem um timeout independente de todos os demais. Todos esses timers podem ser facilmente simulados por software, usando-se um único relógio de hardware que provoca interrupções periódicas. Os timeouts pendentes formam uma lista ligada, com cada nó da lista informando a quantidade de pulsos do relógio até o timer expirar, o quadro que está sendo sincronizado e um ponteiro para o nó seguinte.

[arte: ver original p. 223]

[Dísticos]

[1]Tempo real

10:00:00.0

[2]10:00:00.5

[3]5 1 8 2 6 3 8 2 6 3

[4]Ponteiro para o próximo timeout

Quadro que está sendo sincronizado

Pulsos que faltam

(a) (b)

[F]Figura 3.18

[FL] Simulação de vários timers por software

Para ilustrar como os timers poderiam ser implementados, considere o exemplo da Figura 3.18(a). Suponha que o relógio pulse uma vez a cada 100 ms. Inicialmente, o tempo real é 10:00:00.0 e há três timeouts pendentes, em 10:00:00.5, 10:00:01.3 e 10:00:01.9. Toda vez que o relógio de hardware pulsar, o tempo real será atualizado e o contador de pulsos no início da lista será decrementado. Quando o contador de pulsos for igual a zero, ocorrerá um timeout e o nó será removido da lista, como mostra a Figura 3.18(b). Embora essa organização exija que a lista seja examinada quando *start_timer* ou *stop_timer* for chamado, ela não requer muito trabalho por pulso. No protocolo 5, essas duas rotinas receberam um parâmetro, que indica o quadro a ser sincronizado.

[T3] 3.4.3 Um protocolo que utiliza retransmissão seletiva

O protocolo 5 funciona bem quando há poucos erros, mas se a linha estiver muito ruidosa, ele desperdiçará muita largura de banda com os quadros retransmitidos. Uma estratégia alternativa para lidar com erros é permitir que o receptor aceite e coloque no buffer os quadros subseqüentes a um quadro danificado ou perdido. Esse protocolo não descarta quadros apenas porque um

quadro anterior foi danificado ou perdido.

Nesse protocolo, tanto o transmissor quanto o receptor mantêm uma janela de números de seqüência aceitáveis. O tamanho da janela do transmissor é medido a partir de 0 e atinge um número máximo predefinido, *MAX_SEQ*. Por outro lado, a janela do receptor tem sempre um tamanho fixo e igual a *MAX_SEQ*. O receptor tem um buffer reservado para cada número de seqüência dentro de sua janela fixa. Associado a cada buffer há um bit (*arrived*) que informa se o buffer está cheio ou vazio. Sempre que um quadro chega, seu número de seqüência é verificado pela função *between*, para confirmar se ele se enquadra na janela. Se isso ocorrer e se o quadro ainda não tiver sido recebido, ele será aceito e armazenado. Essa ação é executada sem levar em conta se o quadro contém ou não o próximo pacote esperado pela camada de rede. É óbvio que ele deve ser mantido dentro da camada de enlace de dados e não deve ser repassado à camada de rede, até que todos os quadros de números mais baixos já tenham sido entregues à camada de rede na ordem correta. Um protocolo que utiliza esse algoritmo é apresentado na Figura 3.19.

[arte: ver original da p. 224]

[TD]

```
/* O protocolo 6 (retransmissão seletiva) aceita quadros fora de ordem, mas
repassa pacotes
   para a camada de rede obedecendo à ordem de transmissão. Há um timer
associado a
   cada quadro pendente. Quando o timer expira, apenas o quadro que o contém
   é retransmitido, e não todos os quadros pendentes, como ocorria no protocolo
5. */

#define MAX_SEQ 7                          /* deve ser 2^n - 1 */
```

```
#define NR_BUFS ((MAX_SEQ + 1)/2)

typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready,

ack_timeout} event_type;

#include "protocol.h"

boolean no_nak = true;                    /* nenhuma nak enviada ainda */

seq_nr oldest_frame = MAX_SEQ + 1; /* valor inicial apenas para o simulador */

static boolean between(seq_nr a, seq_nr b, seq_nr c)

{

/* Igual a between do protocolo 5, mas é menor e mais obscuro. */

  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));

}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected,

packet buffer[])

{

/* Monta e envia um quadro de dados, ack ou nak. */

  frame s;                                /* variável de rascunho */


  s.kind = fk;                            /* kind == data, ack ou nak */

  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];

  s.seq = frame_nr;                  /* só tem significado para quadros de dados */

  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);

  if (fk == nak) no_nak = false;  /* uma nak para cada quadro, por favor */

  to_physical_layer(&s);                  /* transmite o quadro */

  if (fk == data) start_timer(frame_nr % NR_BUFS);

  stop_ack_timer();                       /* não precisa de quadro ack separado */

}

void protocol6(void)
```

```
{
  seq_nr ack_expected;                  /* borda inferior da janela do transmissor
*/
  seq_nr next_frame_to_send;            /* borda superior da janela do transmissor
+ 1 */
  seq_nr frame_expected;           /* borda inferior da janela do receptor */
  seq_nr too_far;                       /* borda superior da janela do receptor + 1
*/
  int i;                                /* índice para pool de buffers */
  frame r;                              /* variável de rascunho */
  packet out_buf[NR_BUFS];              /* buffers para o fluxo enviado */
  packet in_buf[NR_BUFS];          /* buffers para o fluxo recebido */
  boolean arrived[NR_BUFS];             /* bitmap de entrada */
  seq_nr nbuffered;                /* número de buffers de saída em uso */
  event_type event;

  enable_network_layer();               /* inicializa */
  ack_expected = 0;                     /* próximo quadro esperado no fluxo
recebido */
  next_frame_to_send = 0;               /* número do próximo quadro a ser
enviado */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                        /* nenhum pacote é inserido no buffer
inicialmente */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
  while (true) {
```

```
    wait_for_event(&event);              /* cinco possibilidades: veja event_type
anterior */
    switch(event) {
      case network_layer_ready: /* aceita, salva e transmite um novo quadro */
            nbuffered = nbuffered + 1; /* expande a janela */
            from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /*
busca novo pacote */
            send_frame(data, next_frame_to_send, frame_expected, out_buf);  /*
transmite o quadro */
            inc(next_frame_to_send); /* avança borda superior da janela */
            break;

      case frame_arrival:           /* chegou um quadro de dados ou de controle */
            from_physical_layer(&r);/* busca quadro recebido da camada física */
            if (r.kind == data) {
                  /* Chegou um quadro sem danos. */
                  if ((r.seq != frame_expected) && no_nak)
                     send_frame(nak, 0, frame_expected, out_buf); else
start_ack_timer();
                  if (between(frame_expected, r.seq, too_far) &&
(arrived[r.seq%NR_BUFS] == false)) {
                        /* Os quadros podem ser aceitos em qualquer ordem. */
                        arrived[r.seq % NR_BUFS] = true;       /* marca buffer como
cheio */
                        in_buf[r.seq % NR_BUFS] = r.info;      /* insere dados no
buffer */
                        while (arrived[frame_expected % NR_BUFS]) {
```

```
                        /* Repassa quadros e avança janela. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);    /* avança borda inferior da
janela do receptor */
                        inc(too_far);           /* avança borda superior da
janela do receptor */
                        start_ack_timer();  /* para ver se é necessária uma ack
separada */
                    }
                }
            }
            if((r.kind==nak) &&
between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
                send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected,
out_buf);

            while (between(ack_expected, r.ack, next_frame_to_send)) {
                nbuffered = nbuffered - 1;   /* trata ack com piggyback */
                stop_timer(ack_expected % NR_BUFS);     /* quadro chegou
intacto */
                inc(ack_expected);      /* avança borda inferior da janela do
transmissor */
            }
            break;
```

```
        case cksum_err:
                if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* quadro
danificado */
                break;

        case timeout:
                send_frame(data, oldest_frame, frame_expected, out_buf);  /*
chegamos ao timeout */
                break;

        case ack_timeout:
                send_frame(ack,0,frame_expected, out_buf);    /* timer de ack expirou;
envia ack */
     }
     if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
  }
} [TN]
```

[F]Figura 3.19

[FL] Um protocolo de janela deslizante que utiliza a retransmissão seletiva

A recepção não seqüencial introduz determinados problemas que não estão presentes em protocolos nos quais os quadros só são aceitos em ordem. Podemos ilustrar melhor o problema com um exemplo. Imagine que haja um número de seqüência de 3 bits, de modo que o transmissor tenha permissão para transmitir até sete quadros antes de ser obrigado a esperar por uma confirmação. Inicialmente, as janelas do transmissor e do receptor são semelhantes às da Figura 3.20(a). No momento, o transmissor envia os quadros de 0 a 6. A janela do

receptor permite que ele aceite qualquer quadro com número de seqüência entre 0 e 6 inclusive. Todos os sete quadros chegam corretamente; assim, o receptor os confirma e avança a janela para permitir a recepção de 7, 0, 1, 2, 3, 4 ou 5, como mostra a Figura 3.20(b). Todos os sete buffers são marcados como vazios.

[arte: ver original p. 226]

[Dísticos]

[1] Transmissor 0 1 2 3 4 5 6 7

Receptor 0 1 2 3 4 5 6 7

(a)

[2] 0 1 2 3 4 5 6 7

0 1 2 3 4 5 6 7

(b)

[3] 0 1 2 3 4 5 6 7

0 1 2 3 4 5 6 7

(c)

[4] 0 1 2 3 4 5 6 7

0 1 2 3 4 5 6 7

(d)

[F]Figura 3.20

[FL] (a) Situação inicial com uma janela de tamanho sete. (b) Depois que sete quadros são enviados e recebidos, mas não confirmados. (c) Situação inicial com uma janela de tamanho quatro. (d) Depois que quatro quadros são enviados e recebidos, mas não confirmados

Nesse ponto ocorre o desastre, na forma de um raio que atinge a central telefônica e apaga todas as confirmações. Mais tarde, o transmissor entra em timeout e retransmite o quadro 0. Quando esse quadro chega ao receptor, é feita

uma conferência para ver se o quadro se ajusta à janela do receptor. Infelizmente, na Figura 3.20(b), o quadro 0 está dentro da nova janela e, assim ele será aceito. O receptor envia uma confirmação com piggyback para o quadro 6, pois os quadros de 0 a 6 foram recebidos.

O transmissor fica feliz em saber que todos os quadros transmitidos chegaram realmente de forma correta; portanto, ele avança sua janela e envia imediatamente os quadros 7, 0, 1, 2, 3, 4 e 5. O quadro 7 será aceito pelo receptor e seu pacote será repassado diretamente à camada de rede. Logo depois, a camada de enlace de dados receptora verifica se já tem um quadro 0 válido, descobre que sim e repassa o pacote incorporado à camada de rede. Conseqüentemente, a camada de rede recebe um pacote incorreto e o protocolo falha.

A essência do problema é que, depois que o receptor avançou a janela, a nova faixa de números de seqüência válidos substituiu a antiga. O próximo lote de quadros poderia ser formado por cópias (se todas as confirmações se perderam) ou de novos quadros (se todas as confirmações foram recebidas). O receptor não tem como distinguir esses dois casos.

A saída desse dilema reside em ter certeza de que, depois que o receptor avança sua janela, não há sobreposição entre esta e a janela original. Para assegurar que não há sobreposição, o tamanho máximo da janela deve ser no máximo igual à metade do intervalo dos números de seqüência, como ocorre nas Figuras 3.20(c) e 3.20(d). Por exemplo, se forem utilizados 4 bits para os números de seqüência, estes irão variar de 0 a 15. Apenas oito quadros não confirmados devem estar pendentes em qualquer instante. Dessa forma, se o receptor só tiver aceito os quadros de 0 a 7 e avançado sua janela para aceitar os quadros de 8 a 15, ele poderá saber sem qualquer dúvida se os quadros subseqüentes são retransmissões (0 a 7) ou novos quadros (8 a 15). Em geral, o tamanho da janela

para o protocolo 6 será (*MAX_SEQ* + 1)/2. Desse modo, para números de seqüência de três bits, o tamanho da janela é quatro.

Uma pergunta interessante é: quantos buffers o receptor deverá ter? De maneira alguma ele aceitará quadros cujos números de seqüência estejam abaixo da borda inferior da janela ou acima da borda superior. Conseqüentemente, o número de buffers necessário é igual ao tamanho da janela, e não ao intervalo dos números de seqüência. No exemplo anterior de um número de seqüência de 4 bits, são necessários oito buffers, numerados de 0 a 7. Quando o quadro i chega, ele é colocado no buffer i mod 8. Observe que apesar de i e (i + 8) mod 8 estarem "competindo" pelo mesmo buffer, eles nunca estão dentro da janela ao mesmo tempo, pois isso implicaria um tamanho de janela de no mínimo 9.

Pela mesma razão, o número de timers necessários é igual ao número de buffers, e não ao tamanho do espaço de seqüência. Efetivamente, existe um timer associado a cada buffer. Quando o timer chega ao seu timeout, o conteúdo do buffer é retransmitido.

No protocolo 5, há uma suposição implícita de que o canal está muito carregado. Quando um quadro chega, nenhuma confirmação é enviada imediatamente. Em vez disso, a confirmação é transportada junto com o próximo quadro de dados a ser enviado. Se o tráfego inverso for leve, a confirmação será retida por um longo período de tempo. Se houver um tráfego intenso em um sentido e nenhum tráfego no outro, apenas pacotes *MAX_SEQ* serão enviados, e então o protocolo será bloqueado; foi por essa razão que tivemos de supor que sempre havia algum tráfego no sentido inverso.

No protocolo 6, esse problema é corrigido. Depois que um quadro de dados seqüencial é recebido, um timer auxiliar é iniciado por *start_ack_timer*. Se nenhum tráfego inverso tiver se apresentado antes do término do intervalo de temporização, um quadro de confirmação separado será enviado. Uma

interrupção provocada pelo timer auxiliar é chamada evento *ack_timeout*. Diante dessa organização, o fluxo de tráfego unidirecional passa a ser possível nesse momento, pois a falta de quadros de dados inversos nos quais as confirmações podem ser transportadas não representa mais um obstáculo. Existe apenas um timer auxiliar e, se *start_ack_timer* for chamado durante o intervalo em que o timer estiver funcionando, ele será reinicializado para um período completo de timeout de confirmação.

E essencial que o timeout associado ao timer auxiliar seja ligeiramente mais curto que o timer utilizado para sincronizar quadros de dados. Essa condição é necessária para assegurar que a confirmação de um quadro corretamente recebido chegue antes de expirar o timer de retransmissão do quadro, de modo que o transmissor não tenha de retransmitir o quadro.

O protocolo 6 utiliza uma estratégia mais eficiente que o protocolo 5 para tratamento de erros. Sempre que tem motivos para suspeitar da ocorrência de um erro, o receptor envia um quadro de confirmação negativa (NAK) de volta ao transmissor. Esse quadro é um pedido de retransmissão do quadro especificado na NAK. Existem dois casos que podem provocar a suspeita do receptor: a chegada de um quadro danificado ou de um quadro diferente do esperado (quadro potencialmente perdido). Para impedir que sejam feitas várias solicitações de retransmissão do mesmo quadro perdido, o receptor deve controlar se já foi enviada uma NAK correspondente a um dado quadro. A variável *no_nak* do protocolo 6 será verdadeira se nenhuma NAK tiver sido enviada ainda para *frame_expected*. Se a NAK for danificada ou perdida, não haverá qualquer prejuízo real pois, com o término do intervalo de timeout, o transmissor irá retransmitir o quadro ausente, de qualquer forma. Se um quadro errado chegar depois que uma NAK tiver sido enviada e perdida, *no_nak* será verdadeira e o timer auxiliar será inicializado. Quando o timer expirar, uma ACK será enviada

para ressincronizar o transmissor com o status atual do receptor.

Em algumas situações, o tempo necessário para que um quadro se propague até o destino, seja processado e tenha a confirmação retornada é (praticamente) constante. Nessas situações, o transmissor pode ajustar seu timer para um tempo ligeiramente maior que o intervalo normal esperado entre o envio de um quadro e a recepção de sua confirmação. Entretanto, se o tempo for bastante variável, o transmissor terá de optar entre ajustar o intervalo com um valor pequeno (e arriscar-se a retransmissões desnecessárias) ou ajustá-lo com um valor grande (e ficar ocioso por um longo período após um erro).

Ambas as opções desperdiçam largura de banda. Se o tráfego inverso for esporádico, o tempo antes da confirmação será irregular, sendo mais curto quando houver tráfego inverso e mais longo quando não houver. O tempo de processamento variável dentro do receptor também pode ser um problema nesse caso. Em geral, sempre que o desvio padrão do intervalo de confirmação é pequeno em comparação com o próprio intervalo, o timer pode ser ajustado "com maior rigor" e as NAKs deixam de ser úteis. Caso contrário, o timer pode ser ajustado "mais livremente", a fim de evitar retransmissões desnecessárias; porém, as NAKs podem acelerar bastante a retransmissão de quadros perdidos ou danificados.

Um problema intimamente relacionado com o uso de timeouts e NAKs é a questão de determinar o quadro que provocou um timeout. No protocolo 5, ele é sempre *ack_expected*, porque é sempre o mais antigo. No protocolo 6, não há qualquer forma trivial para determinar o quadro que chegou ao timeout. Imagine que os quadros de 0 a 4 tenham sido transmitidos, significando que a lista de quadros pendentes é 01234, na ordem do mais antigo para o mais recente. Agora, imagine que o quadro 0 chegue ao timeout, que 5 (um novo quadro) seja transmitido, 1 e 2 cheguem ao timeout e 6 (outro quadro novo) seja transmitido.

Nesse ponto, a lista de quadros pendentes será 3405126, na ordem do mais antigo para o mais recente. Se todo o tráfego de chegada (isto é, quadros que transportam confirmações) for perdido durante algum tempo, esses sete quadros pendentes chegarão ao timeout nessa ordem.

Para evitar que o exemplo fique ainda mais complicado do que já está, não mostramos a administração do timer. Em vez disso, consideramos apenas que a variável *oldest_frame* está ativa no momento do timeout para indicar o quadro que chegou ao timeout.

[T2] 3.5 Verificação de protocolos

Os protocolos realistas e os programas que os implementam em geral são bastante complicados. Conseqüentemente, várias pesquisas foram realizadas na tentativa de descobrir técnicas matemáticas formais para a especificação e a verificação de protocolos. Nas seções a seguir, serão apresentados alguns modelos e técnicas. Apesar de estarmos analisando esses modelos e técnicas no contexto da camada de enlace de dados, eles também se aplicam a outras camadas.

[T3] 3.5.1 Modelos de máquinas de estados finitos

Um conceito fundamental utilizado em vários modelos de protocolos é o de **máquina de estados finitos.** Com essa técnica, cada máquina de protocolo (isto é, o transmissor ou o receptor) está sempre em um estado específico a cada instante. Seu estado consiste em todos os valores de suas variáveis, inclusive o contador de programa.

Na maioria dos casos, um grande número de estados pode ser agrupado para fins de análise. Por exemplo, considerando o receptor no protocolo 3, é possível abstrair dois estados importantes dentre todos os outros possíveis: a espera do

quadro 0 ou a espera do quadro 1. Todos os outros estados podem ser considerados transientes, simplesmente etapas que levam a um dos estados principais. Em geral, os estados são escolhidos como os instantes em que a máquina de protocolo está esperando pela ocorrência do evento seguinte [isto é, executando a chamada de procedimento *wait* (*event*) em nossos exemplos]. Nesse ponto, o estado da máquina de protocolo é completamente determinado pelos estados de suas variáveis. O número de estados é então 2^n, onde n é o número de bits necessários para representar todas as variáveis combinadas.

O estado do sistema completo é a combinação de todos os estados das duas máquinas de protocolo e do canal. O estado do canal é determinado por seu conteúdo. Utilizando o protocolo 3 novamente como exemplo, o canal tem quatro estados possíveis: um quadro zero ou um quadro um que se move do transmissor para o receptor, um quadro de confirmação se deslocando em sentido contrário, ou um canal vazio. Se o transmissor e o receptor forem modelados com dois estados cada um, o sistema completo terá 16 estados distintos.

Aqui, vale a pena uma ressalva sobre o estado do canal. Obviamente, o conceito da permanência de um quadro "no canal" é uma abstração. Na realidade, queremos dizer que um quadro possivelmente foi recebido, mas ainda não foi processado no destino. Um quadro permanece "no canal" até a máquina de protocolo executar *FromPhysicalLayer* e processá-lo.

A partir de cada estado, há zero ou mais **transições** possíveis para outros estados. As transições ocorrem quando algum evento acontece. No caso de uma máquina de protocolo, pode ocorrer uma transição quando um quadro é enviado, quando um quadro chega, quando um timer expira, quando ocorre uma interrupção etc. No caso do canal, os eventos típicos são a inserção de um novo quadro no canal por uma máquina de protocolo, a entrega de um quadro a uma máquina de protocolo ou a perda de um quadro devido a ruído. Dada uma

descrição completa das máquinas de protocolo e das características do canal, é possível traçar um grafo orientado que mostra todos os estados como nós e todas as transições como arcos orientados.

Um estado específico é designado como o **estado inicial**. Esse estado corresponde à descrição do sistema quando inicia a execução ou em algum ponto de partida conveniente logo após esse instante. A partir do estado inicial, alguns, ou talvez todos os outros estados, podem ser alcançados por uma seqüência de transições. Usando técnicas conhecidas da teoria de grafos (por exemplo, o cálculo do fechamento transitivo de um grafo), é possível determinar os estados que serão acessíveis e os que não serão. Essa técnica é denominada **análise de acessibilidade** (Lin *et al.*, 1987). Essa análise pode ser útil para determinar se um protocolo está correto ou não.

Formalmente, um modelo de máquina de estados finitos de um protocolo pode ser considerado uma quádrupla (*S*, *M*, *I*, *T*), onde:

S é o conjunto de estados em que os processos e o canal podem se encontrar.

M é o conjunto de quadros que podem ser intercambiados pelo canal.

I é o conjunto de estados iniciais dos processos.

T é o conjunto de transições entre estados.

No início da contagem do tempo, todos os processos se encontram em seus estados iniciais. Então, os eventos começam a acontecer, como quadros que se tornam disponíveis para transmissão ou timers que são desativados. Cada evento pode fazer com que um dos processos ou o canal realize uma ação e mude para um novo estado. Enumerando cuidadosamente cada sucessor possível para cada estado, é possível criar o grafo de acessibilidade e analisar o protocolo.

A análise de acessibilidade pode ser utilizada para detectar uma variedade de erros na especificação do protocolo. Por exemplo, se for possível ocorrer um

determinado quadro em um certo estado e a máquina de estados finitos não informar que ação deve ser executada, a especificação estará errada (incompleta). Se houver um conjunto de estados do qual não seja possível sair nem obter qualquer progresso (isto é, não seja possível receber mais quadros corretos), teremos outro erro (impasse). Um erro menos grave é a especificação de protocolo que informa como tratar um evento em um estado no qual ele não pode ocorrer (transição extrínseca). Outros erros também podem ser detectados.

Como exemplo de um modelo de máquina de estados finitos, considere a Figura 3.21(a). Esse grafo corresponde ao protocolo 3, conforme descrito anteriormente: cada máquina de protocolo tem dois estados, enquanto o canal tem quatro estados. Há um total de 16 estados, e nem todos podem ser alcançados a partir do estado inicial. Os que não podem ser alcançados não são mostrados na figura. Os erros de totais de verificação também são ignorados aqui, por simplicidade.

Cada estado é rotulado com três caracteres, *SRC*, onde *S* é 0 ou 1 e corresponde ao quadro que o transmissor está tentando enviar; *R* também é 0 ou 1 e corresponde ao quadro esperado pelo receptor; e *C* é 0, 1, *A* ou vazio (—) e corresponde ao estado do canal. Nesse exemplo, o estado inicial escolhido foi (000). Em outras palavras, o transmissor acabou de enviar o quadro 0, o receptor espera receber o quadro 0, e o quadro 0 está no canal nesse momento.

[arte: ver original p. 231]

[Dísticos]

[1](a)

[2]

Transição	Quem a executa?
0	—
1	R
2	S

3 R

4 S

5 R

6 R

7 S

8 S

[3]

Quadro aceito Quadro emitido

(quadro perdido)

0 A

A 1

1 A

A 0

0 A

1 A

(timeout) 0

(timeout) 1

(b)

[4]

Para a camada de rede

—

Sim

—

Sim

—

Não

Não

—

—

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 3.21

[FL] (a) Diagrama de estados para o protocolo 3. (b) Transições

Nove tipos de transições estão ilustrados na Figura 3.21. A transição 0 consiste na perda do conteúdo do canal. A transição 1 apresenta o canal entregando corretamente o pacote 0 ao receptor, com o receptor alterando seu estado para esperar o quadro 1 e emitir uma confirmação. A transição 1 também corresponde ao momento em que o receptor entrega o pacote 0 à camada de rede. As demais transições estão listadas na Figura 3.21(b). Não foi mostrada a chegada de um quadro com um erro no total de verificação, porque isso não altera o estado (no protocolo 3).

Durante a operação normal, as transições 1, 2, 3 e 4 são repetidas continuamente em ordem. Em cada ciclo, são entregues dois pacotes, levando o transmissor de volta ao estado inicial em que tenta enviar um novo quadro com número de seqüência 0. Se perder o quadro 0, o canal fará uma transição do estado (000) para o estado (00–). Posteriormente, o transmissor chegará ao timeout (transição 7), e o sistema voltará para (000). A perda de uma confirmação é mais complicada e exige duas transições, 7 e 5, ou 8 e 6, para reparar o dano.

Uma das propriedades que um protocolo com um número de seqüência de 1 bit deve ter é que, independente da seqüência de eventos que ocorrer, o receptor

nunca entregará dois pacotes ímpares sem um pacote par interveniente e vice-versa. Observando o grafo da Figura 3.21, verificamos que essa exigência pode ser estabelecida mais formalmente como "não deve existir qualquer caminho a partir do estado inicial no qual duas ocorrências da transição 1 aconteçam sem uma ocorrência da transição 3 entre elas, ou vice-versa". Na figura é possível observar que, sob esse aspecto, o protocolo está correto.

Outra exigência semelhante é que não exista qualquer caminho em que o transmissor mude de estado duas vezes (por exemplo, de 0 para 1 e novamente para 0), enquanto o estado do receptor permanece constante. Se houvesse um caminho desse tipo, na seqüência de eventos correspondente, dois quadros seriam irremediavelmente perdidos, sem que o receptor percebesse. A seqüência de pacotes entregue teria um hiato de dois pacotes.

Outra propriedade importante de um protocolo é a ausência de impasses. Um **impasse** (deadlock) é uma situação em que o protocolo não consegue progredir (isto é, entregar pacotes à camada de rede), não importando a seqüência de eventos que aconteça. Em termos do modelo de grafo, um impasse se caracteriza pela existência de um subconjunto de estados acessível a partir do estado inicial e que tem duas propriedades:

1. Não há qualquer transição fora do subconjunto.
2. No subconjunto não há transições que dêem continuidade ao processo.

Quando chega a uma situação de impasse, o protocolo permanece nessa situação para sempre. Mais uma vez, observando o grafo, fica fácil perceber que não ocorrem impasses no protocolo 3.

[T3] 3.5.2 Modelos de rede de Petri

A máquina de estados finitos não é a única técnica para a especificação formal de protocolos. Nesta seção, será descrita uma técnica bem diferente, a **rede de Petri**

(Danthine, 1980). Uma rede de Petri tem quatro elementos básicos: lugares, transições, arcos e símbolos (tokens). Um **lugar** representa um estado no qual o sistema (ou parte dele) pode estar. A Figura 3.22 mostra uma rede de Petri com dois lugares, *A* e *B*, indicados como círculos. No momento, o sistema se encontra no estado *A*, indicado pelo **símbolo** (ponto escuro) no lugar *A*. Uma **transição** é indicada por uma barra horizontal ou vertical. Cada transição tem zero ou mais **arcos de entrada**, cuja origem são seus lugares de entrada, e zero ou mais **arcos de saída**, que partem em direção a seus lugares de saída.

[arte: ver original p. 232]

[Dísticos]

[1] A 1 B 2

[F]Figura 3.22

[FL] Uma rede de Petri com dois lugares e duas transições

Uma transição está **ativa** se houver pelo menos um símbolo de entrada em cada um de seus lugares de entrada. Qualquer transição ativa pode ser **disparada** à vontade, removendo um símbolo de cada lugar de entrada e depositando um símbolo em cada lugar de saída. Se o número de arcos de entrada e de saída for diferente, os símbolos não serão conservados. Se duas ou mais transições estiverem ativas, qualquer uma das duas poderá ser disparada. A escolha da transição a ser disparada é indeterminada, e esse é o motivo pelo qual as redes de Petri são úteis para a modelagem de protocolos. A rede de Petri da Figura 3.22 é determinística e pode ser utilizada para representar qualquer processo de duas fases (por exemplo, o comportamento de um bebê: comer, dormir, comer, dormir e assim por diante). A exemplo do que ocorre com todas as ferramentas de modelagem, os detalhes desnecessários são suprimidos.

[arte: ver original p. 233]

Rejeita 1

Estado do receptor

[F]Figura 3.23

[FL] Um modelo de rede de Petri correspondente ao protocolo 3

A Figura 3.23 mostra o modelo de rede de Petri referente à Figura 3.12. Ao contrário do modelo de máquina de estados finitos, não há estados compostos nesse caso; o estado do transmissor, o estado do canal e o estado do receptor são representados separadamente. As transições 1 e 2 correspondem à transmissão do quadro 0 pelo transmissor, em condições normais, e a um timeout, respectivamente. As transições 3 e 4 são análogas, mas se referem ao quadro 1. As transições 5, 6 e 7 correspondem à perda do quadro 0, de uma confirmação e do quadro 1, respectivamente. As transições 8 e 9 ocorrem quando um quadro de dados com o número de seqüência errado chega ao receptor. As transições 10 e 11 representam a chegada ao receptor do quadro seguinte e sua entrega à camada de rede.

As redes de Petri podem ser utilizadas para detectar falhas de protocolos, de maneira semelhante à utilização das máquinas de estados finitos. Por exemplo, se alguma seqüência de disparo tivesse incluído a transição 10 duas vezes sem a interveniência da transição 11, o protocolo estaria incorreto. O conceito de impasse em uma rede de Petri também é semelhante ao da máquina de estados finitos.

As redes de Petri podem ser representadas em uma forma algébrica que se assemelha a uma gramática. Cada transição contribui para uma regra da gramática. Cada regra especifica os lugares de entrada e saída da transição. Tendo em vista que a Figura 3.23 tem 11 transições, sua gramática tem 11 regras, numeradas de 1 a 11, e cada uma corresponde à transição que tem o

mesmo número. A gramática referente à rede de Petri da Figura 3.23 é:

1: BD → AC

2: A → A

3: AD → BE

4: B → B

5: C →

6: D →

7: E →

8: CF → DF

9: EG → DG

10: CG → DF

11: EF → DG

É interessante observar que conseguimos reduzir um protocolo complexo a 11 regras gramaticais simples que podem ser manipuladas com facilidade por um programa de computador.

O estado atual da rede de Petri é representado como um conjunto não ordenado de lugares, cada um representado no conjunto tantas vezes quantos são seus símbolos. Qualquer regra na qual os lugares do lado esquerdo estão presentes pode ser disparada, removendo esses lugares do estado atual e adicionando seus lugares de saída ao estado atual. A marcação da Figura 3.23 é *ACG* (isto é, *A*, *C* e *G* têm um símbolo cada um). Conseqüentemente, as regras 2, 5 e 10 estão todas ativas e qualquer delas pode ser aplicada, levando a um novo estado (possivelmente com a mesma marcação do estado original). Em contraste, a regra 3 (*AD* → *BE*) não pode ser aplicada, porque *D* não está marcado.

[T2] 3.6 Exemplos de protocolos de enlace de dados

Nas próximas seções, serão examinados diversos protocolos de enlace de dados

muito utilizados. O primeiro deles, denominado HDLC, é um protocolo clássico orientado a bits, cujas variantes foram utilizadas durante décadas em muitas aplicações. O segundo, chamado PPP, é o protocolo de enlace de dados utilizado para conectar computadores domésticos à Internet.

[T3] 3.6.1 HDLC — High-level Data Link Control

Nesta seção, examinaremos um grupo de protocolos intimamente relacionados que, apesar de um pouco antigos, continuam sendo bastante utilizados. Todos eles são derivados do protocolo de enlace de dados utilizado primeiro no mundo dos computadores de grande porte da IBM: o protocolo **SDLC (Synchronous Data Link Control — controle de enlace de dados síncrono).** Depois de desenvolver o SDLC, a IBM o submeteu ao ANSI e à ISO para aceitação como um padrão nos Estados Unidos e no mundo inteiro, respectivamente. O ANSI o modificou, transformando-o no **ADCCP (Advanced Data Communication Control Procedure — procedimento de controle de comunicação de dados avançado)**, e a ISO o alterou, para transformá-lo no **HDLC (High-level Data Link Control — controle de enlace de dados de alto nível).** Depois disso, o CCITT adotou e modificou o HDLC e o transformou em seu **LAP (Link Access Procedure — procedimento de acesso de enlace)**, como parte do padrão de interface de rede X.25. Porém, mais tarde, o CCITT modificou o padrão novamente e passou a chamá-lo **LAPB**, a fim de torná-lo mais compatível com uma versão posterior do HDLC. A característica mais interessante dos padrões é que há muitos deles para se escolher. Além disso, se não gostar de nenhum, você poderá simplesmente esperar pelo modelo do próximo ano.

Esses protocolos se baseiam nos mesmos princípios. Todos são orientados a bits, e todos utilizam a técnica de inserção de bits para transparência de dados. Eles diferem apenas em pequenos e irritantes detalhes. A discussão dos protocolos

orientados a bits apresentada a seguir foi elaborada como uma introdução geral. Para obter detalhes específicos a respeito de qualquer protocolo, consulte a definição apropriada.

Todos os protocolos orientados a bits utilizam a estrutura de quadro apresentada na Figura 3.24. O campo *Endereço* é importante principalmente nas linhas com vários terminais, onde ele é utilizado para identificar um dos terminais. No caso de linhas ponto aponto, às vezes esse campo é utilizado para fazer distinção entre comandos e respostas.

[arte: ver original p. 235]

[Dísticos]

[1] Bits 8 8 8 ≥ 0 16 8

[2] 01111110 Endereço Controle Dados Total de verificação 01111110

[F]Figura 3.24

[FL] Formato de quadro para protocolos orientados a bits

O campo *Controle* é usado para números de seqüência, confirmações e outras finalidades, como será discutido a seguir.

O campo *Dados* pode conter qualquer informação. Ele pode ser arbitrariamente longo, embora a eficiência do total de verificação diminua com o aumento do comprimento do quadro, devido à maior probabilidade de ocorrerem vários erros em rajada.

O campo *Total de verificação* é uma variação do código de redundância cíclica que utiliza a técnica examinada na Seção 3.2.2.

O quadro é delimitado por outra seqüência de flag (01111110). Nas linhas ponto a ponto ociosas, as seqüências de flags são transmitidas de forma contínua. O quadro mínimo contém três campos e totaliza 32 bits, excluindo os flags de cada

extremidade.

Existem três tipos de quadros: **Quadro de informação**, **Quadro supervisor** e **Quadro não numerado.** O conteúdo do campo *Controle* para esses três tipos de quadros é apresentado na Figura 3.25. O protocolo utiliza uma janela deslizante, com um número de seqüência de 3 bits. A qualquer momento, pode haver até sete quadros não confirmados pendentes. O campo *Seq* da Figura 3.25(a) é o número de seqüência do quadro. O campo *Próximo* é uma confirmação transportada por piggyback. Entretanto, todos os protocolos aderem à convenção de, em vez de transportar o número do último quadro recebido corretamente, utilizar o número do primeiro quadro ainda não recebido (isto é, o próximo quadro esperado). A escolha entre utilizar o último quadro recebido ou o próximo quadro esperado é arbitrária; não importa que convenção é adotada, desde que ela seja utilizada de forma coerente.

[arte: ver original p. 236]

[Dísticos]

[1]

Bits	1	3	1	3
(a)	0	Seq	P/F	Próximo
(b)	1 0	Tipo	P/F	Próximo
(c)	1 1	Tipo	P/F	Modificador

[F]Figura 3.25

[FL] Campo de controle de (a) um quadro de informação, (b) um quadro supervisor e (c) um quadro não numerado

O bit *P/F* representa *Poll/Final*. Ele é utilizado quando um computador (ou concentrador) está consultando um grupo de terminais. Quando utilizado como *P*, o computador está convidando o terminal a enviar os dados. Todos os quadros

enviados pelo terminal, com exceção do quadro final, têm o bit *P*/*F* definido como *P*. O quadro final é definido como *F*.

Em alguns protocolos, o bit *P/F* é utilizado para forçar a outra máquina a enviar imediatamente um quadro supervisor, em vez de aguardar o tráfego inverso para inserir nele as informações da janela. O bit também tem alguns usos menos importantes relacionados aos quadros não numerados.

Os diversos tipos de quadros supervisores se distinguem pelo campo *Tipo*. O Tipo 0 é um quadro de confirmação (denominado oficialmente RECEIVE READY) usado para indicar o próximo quadro esperado. Esse quadro é utilizado quando não há tráfego inverso que permita o uso do piggybacking.

O Tipo 1 é um quadro de confirmação negativa (denominado oficialmente REJECT). Ele é utilizado para indicar a detecção de um erro de transmissão. O campo *Próximo* indica o primeiro quadro da seqüência não recebido corretamente (isto é, o quadro a ser retransmitido). O transmissor é solicitado a retransmitir todos os quadros pendentes a partir de *Próximo*. Essa estratégia é semelhante ao nosso protocolo 5, e não ao protocolo 6.

O Tipo 2 é RECEIVE NOT READY. Ele confirma todos os quadros até (mas não incluindo) *Próximo*, exatamente como RECEIVE READY, mas solicita que o transmissor interrompa o envio de quadros. RECEIVE NOT READY tem como objetivo informar a existência de determinados problemas temporários com o receptor, tais como a insuficiência de buffers, e não representa uma alternativa para o controle de fluxo de janela deslizante. Quando a condição tiver sido corrigida, o receptor enviará RECEIVE READY, REJECT ou certos quadros de controle.

O Tipo 3 é SELECTIVE REJECT, que solicita a retransmissão apenas do quadro especificado. Nesse sentido, ele se assemelha mais ao nosso protocolo 6 que ao protocolo 5 e, portanto, é mais útil quando o tamanho da janela do transmissor é

menor ou igual à metade do tamanho do espaço de seqüência. Dessa forma, se um receptor desejar armazenar quadros fora de seqüência no buffer para uma possível utilização futura, ele poderá forçar a retransmissão de qualquer quadro específico utilizando SELECTIVE REJECT. O HDLC e o ADCCP permitem esse tipo de quadro, mas o SDLC e o LAPB não o permitem (isto é, não há SELECTIVE REJECT), e os quadros do Tipo 3 são indefinidos.

A terceira classe de quadro é o quadro não numerado que, às vezes, é utilizado para fins de controle, mas que também pode transportar dados quando é utilizado o serviço não confiável sem conexão. Os diversos protocolos orientados a bits diferem consideravelmente nesse ponto, ao contrário dos outros dois tipos, nos quais eles são quase idênticos. Há cinco bits disponíveis para indicar o tipo de quadro, mas nem todas as 32 possibilidades são utilizadas.

Todos os protocolos dispõem de um comando, DISC (de DISConnect — Desconectar), que permite a uma máquina anunciar que está se desativando (por exemplo, para manutenção preventiva). Eles também oferecem um comando que permite a uma máquina que acabou de se conectar anunciar sua presença e forçar todos os números de seqüência de volta a zero. Esse comando é denominado SNRM (Set Normal Response Mode). Infelizmente, o "modo normal de resposta" é tudo, menos normal. Trata-se de um modo desbalanceado (isto é, assimétrico) em que um extremo da linha é o mestre e o outro é o escravo. O SNRM data de uma época em que a comunicação de dados significava um terminal burro que se comunicava com um enorme computador host, algo claramente assimétrico. Para tornar o protocolo mais adequado quando os dois parceiros são iguais, o HDLC e o LAPB têm um comando adicional, o SABM (Set Asynchronous Balanced Mode), que restabelece a linha e declara as duas partes como equivalentes. Eles também têm comandos SABME e SNRME, que são iguais aos comandos SABM e SNRM, respectivamente, exceto pelo fato de ativarem um

formato de quadro estendido que utiliza números de seqüência de 7 bits em lugar de números de seqüência de 3 bits.

Um terceiro comando fornecido por todos os protocolos é o FRMR (FRaMe Reject), utilizado para indicar a chegada de um quadro com total de verificação correto, mas de semântica impossível. Exemplos de semântica impossível são um quadro supervisor do tipo 3 em LAPB, um quadro com menos de 32 bits, um quadro de controle inválido e uma confirmação de um quadro que estava fora da janela etc. Os quadros de FRMR contêm um campo de dados de 24 bits que informa o que estava errado com o quadro. Dentre esses dados, estão o campo de controle do quadro com erros, os parâmetros da janela e um conjunto de bits que indica erros específicos.

Os quadros de controle podem estar perdidos ou danificados, da mesma forma que os quadros de dados, e assim eles também devem ser confirmados. Um quadro de controle especial, denominado UA, (Unnumbered Acknowledgment) é fornecido para esse fim. Como apenas um quadro de controle pode estar pendente, nunca haverá qualquer ambigüidade em relação ao quadro de controle que está sendo confirmado.

Os quadros de controle restantes se referem à inicialização, ao polling e a relatórios de status. Também existe um quadro de controle que pode conter informações arbitrárias, o UI (Unnumbered Information). Esses dados não são repassados à camada de rede, mas se destinam à própria camada de enlace de dados do receptor.

Apesar de sua ampla utilização, o HDLC está longe de ser perfeito. Uma discussão sobre uma variedade de problemas associados a ele pode ser encontrada em (Fiorini *et al.*,1994).

[T3] 3.6.2 A camada de enlace de dados na Internet

A Internet consiste em máquinas individuais (hosts e roteadores) e na infra-estrutura de comunicação que as conecta. Dentro de um único prédio, as LANs são bastante utilizadas para interconexões, mas grande parte da infra-estrutura geograficamente distribuída é construída a partir de linhas privadas ponto a ponto. O Capítulo 4 tratará das LANs; neste capítulo, examinaremos os protocolos de enlace de dados utilizados em linhas ponto a ponto na Internet.

Na prática, a comunicação ponto a ponto é utilizada principalmente em duas situações. Na primeira delas, milhares de organizações têm uma LAN ou mais, cada uma com um determinado número de hosts (computadores pessoais, estações de trabalho, servidores etc.) e um roteador (ou uma ponte, de funcionalidade semelhante). Com freqüência, os roteadores são interconectados por uma LAN de backbone. Em geral, todas as conexões com o mundo exterior passam por um ou dois roteadores que têm linhas privadas (também chamadas linhas dedicadas) ponto a ponto com roteadores distantes. São esses roteadores e suas linhas privadas que compõem as sub-redes de comunicação, nas quais a Internet se baseia.

A segunda situação em que as linhas ponto a ponto executam uma função importante na Internet diz respeito aos milhões de indivíduos que estabelecem conexões domésticas com a Internet utilizando modems e linhas telefônicas com acesso por discagem. Geralmente, o PC doméstico do usuário estabelece uma conexão com o roteador de um provedor de serviços da Internet, e depois atua como um host da Internet completo. Esse método de operação não difere de ter uma linha privada entre o PC e o roteador, exceto pelo fato de a conexão ser encerrada quando o usuário finaliza a sessão. Um PC doméstico que se conecta a um provedor de serviços da Internet está ilustrado na Figura 3.26. Mostramos o modem externo ao computador para enfatizar sua função , mas os computadores modernos têm modems internos.

[arte: ver original p. 238]

[Dísticos]

[1]Casa do usuário

[2]PC

[3]Processo cliente que utiliza TCP/IP

[4]Modem

[5]Linha telefônica com acesso por discagem

[6]Conexão TCP/IP que utiliza PPP

[7]Instalações do provedor da Internet

[8]Modems

[9]Roteador

[10]Processo de roteamento

[F]Figura 3.26

[FL] Um computador pessoal doméstico que atua como um host da Internet

Tanto para a conexão de linha privada entre roteadores quanto para a conexão com acesso por discagem entre o host e o roteador, é necessário o uso de um protocolo de enlace de dados ponto a ponto na linha para cuidar do enquadramento, do controle de erros e de outras funções da camada de enlace de dados que estudamos neste capítulo. O único protocolo utilizado na Internet é o PPP. Vamos examiná-lo agora.

[T4] PPP — Point-to-Point Protocol

A Internet precisa de um protocolo ponto a ponto para diversos fins, inclusive para cuidar do tráfego de roteador para roteador e de usuário doméstico para ISP (provedor de serviços da Internet). Esse protocolo é o **PPP (Point-to-Point Protocol — protocolo ponto a ponto)**, definido na RFC 1661 e mais elaborado em

várias outras RFCs (por exemplo, as RFCs 1662 e 1663). O PPP trata da detecção de erros, aceita vários protocolos, permite que endereços IP sejam negociados em tempo de conexão, permite a autenticação e inclui muitas outras características.

O PPP dispõe de três recursos:

1. Um método de enquadramento que delineia de forma não ambígua o fim de um quadro e o início do quadro seguinte. O formato do quadro também lida com a detecção de erros.
2. Um protocolo de controle de enlace usado para ativar linhas, testá-las, negociar opções e desativá-las novamente quando não forem mais necessárias. Esse protocolo é denominado **LCP (Link Control Protocol — protocolo de controle de enlace).** Ele admite circuitos síncronos e assíncronos, e também codificações orientadas a bytes e a bits.
3. Uma maneira de negociar as opções da camada de rede de modo independente do protocolo da camada de rede a ser utilizado. O método escolhido deve ter um **NCP (Network Control Protocol — protocolo de controle de rede)** diferente para cada camada de rede aceita.

Para verificar como esses itens se encaixam uns com os outros, considere a situação típica em que um usuário doméstico se conecta a um provedor de serviços da Internet para transformar um PC doméstico em um host temporário da Internet. Primeiro, o PC chama o roteador do provedor por meio de um modem. Depois que o modem do roteador atende ao telefone e estabelece uma conexão física, o PC envia ao roteador uma série de pacotes LCP no campo de carga útil de um ou mais quadros PPP. Esses pacotes e suas respostas selecionam os parâmetros PPP a serem utilizados.

Quando todos esses parâmetros estão corretamente definidos de comum acordo, uma série de pacotes NCP é enviada para configurar a camada de rede. Em geral, o PC quer executar uma pilha de protocolos TCP/IP, e assim necessita de um

endereço IP. Como não há endereços IP suficientes, normalmente cada provedor da Internet obtém um bloco de endereços e, em seguida, atribui dinamicamente um endereço a cada PC recém-conectado durante sua sessão de login. Se tiver *n* endereços IP, um provedor poderá ter até *n* máquinas conectadas simultaneamente, mas sua base total de clientes poderá estar muito acima desse número. O NCP para o IP atribui os endereços IP.

Nesse momento, o PC passa a ser um host da Internet e pode enviar e receber pacotes IP, da mesma forma que os hosts fisicamente conectados. Quando o usuário termina, o NCP é utilizado para desativar a conexão da camada de rede e liberar o endereço IP. Em seguida, o LCP encerra a conexão da camada de enlace de dados. Finalmente, o computador solicita que o modem desligue o telefone, liberando a conexão da camada física.

O formato de quadro PPP foi definido de modo a ter uma aparência semelhante ao formato de quadro HDLC, pois não há motivo algum para a definição de um novo padrão. A principal diferença entre o PPP e o HDLC é que o primeiro é orientado a caracteres, e não a bits. Especificamente, o PPP utiliza a técnica de inserção de bytes em linhas de discagem por modem; portanto, todos os quadros representam um número inteiro de bytes. Não é possível enviar um quadro formado por 30,25 bytes, como ocorre com o HDLC. Os quadros PPP não só podem ser enviados por linhas telefônicas de acesso por discagem, mas também podem ser enviados por linhas SONET ou por verdadeiras linhas HDLC orientadas a bits (por exemplo, para as conexões entre roteadores). A Figura 3.27 mostra o formato do quadro PPP.

[arte: ver original p. 240]

[Dísticos]

[1] Bytes 1 1 1 1 ou 2 Variável 2 ou 4 1

[2] Flag Endereço Controle Protocolo Carga útil Total de

verificação Flag

01111110 11111111 00000011 01111110

[F]Figura 3.27

[FL] O formato completo do quadro PPP para a operação no modo não numerado

Todos os quadros PPP começam pelo byte de flag padrão do HDLC (01111110), que é complementado por inserção de bytes se ocorrer dentro do campo de carga útil. Em seguida, temos o campo *Endereço*, que sempre é definido como o valor binário 11111111, indicando que todas as estações devem aceitar o quadro. A utilização desse valor evita o problema da necessidade de atribuição de endereços de enlace de dados.

O campo *Controle* é exibido após o campo *Endereço* e seu valor padrão é 00000011. Esse valor indica um quadro não numerado. Em outras palavras, o PPP não oferece uma transmissão confiável com o uso de números de seqüência e confirmações como o padrão. Em ambientes ruidosos, como em redes sem fio, pode ser utilizada a transmissão confiável que emprega o modo numerado. Os detalhes exatos são definidos na RFC 1663 mas, na prática, raramente ele é utilizado.

Como os campos *Endereço* e *Controle* são sempre constantes na configuração padrão, o LCP fornece o mecanismo necessário para que as duas partes negociem uma opção que os omita totalmente e que economize 2 bytes por quadro.

O quarto campo do quadro PPP é o campo *Protocolo*. Sua tarefa é informar o tipo de pacote que se encontra no campo *Carga útil*. Os códigos são definidos para representar os protocolos LCP, NCP, IP, IPX, AppleTalk e outros. Os protocolos que começam por um bit 0 são os protocolos da camada de rede, como o IP, o IPX, o OSI, o CLNP, o XNS. Aqueles que começam por um bit 1 são utilizados na negociação de outros protocolos. Entre eles estão incluídos o LCP e um NCP

diferente para cada protocolo da camada de rede admitido. O tamanho padrão do campo *Protocolo* é 2 bytes, mas é possível negociar uma redução para 1 byte, utilizando-se o LCP.

O campo *Carga útil* tem comprimento variável, podendo se estender até o tamanho máximo negociado. Se o comprimento não for negociado com o uso do LCP durante a configuração da linha, será empregado um comprimento padrão de 1.500 bytes. Poderá haver um preenchimento logo após a carga útil, caso seja necessário.

Depois do campo *Carga Útil*, temos o campo *Total de verificação*, que normalmente tem 2 bytes, embora seja possível negociar um total de verificação de 4 bytes.

Em suma, o PPP é um mecanismo de enquadramento multiprotocolo, adequado para a utilização em modems, em linhas seriais de bits HDLC, na SONET e em outras camadas físicas. Ele aceita a detecção de erros, a negociação de opções, a compactação de cabeçalhos e, opcionalmente, a transmissão confiável com o uso de um formato de quadro do tipo HDLC.

Agora, vamos deixar o estudo do formato do quadro PPP para examinar a maneira como as linhas são ativadas e desativadas. O diagrama (simplificado) representado na Figura 3.28 mostra as fases pelas quais uma linha passa ao ser ativada, utilizada e desativada novamente. Essa seqüência se aplica tanto às conexões de modem quanto às conexões entre roteadores.

[arte: ver original p. 241]

[Dísticos]

[1]Portadora detectada Os dois lados concordam quanto às opções

Autenticação bem-sucedida

[2] Establish Authenticate

Falhou

[3]Dead Network

Falhou

[4] Terminate Open

[5]Portadora desativada Concluído Configuração NCP

[F]Figura 3.28

[FL] Um diagrama simplificado de fases para ativar e desativar uma linha

O protocolo começa com a linha no estado *DEAD*, o que significa que não há nenhuma portadora da camada física presente e não existe qualquer conexão da camada física. Depois de estabelecida a conexão física, a linha passa para a fase *ESTABLISH*. Nesse ponto, começa a negociação de opções do LCP que, se for bem-sucedida, levará à fase *AUTHENTICATE*. Agora, as duas partes poderão verificar suas identidades mutuamente, se desejarem. Quando a fase *NETWORK* é alcançada, o protocolo NCP apropriado é invocado para configurar a camada de rede. Se a configuração for bem-sucedida, a fase *OPEN* é alcançada e o transporte de dados pode ser feito. Quando o transporte de dados é concluído, a linha entra na fase *TERMINATE* e, de lá, volta a *DEAD* quando a portadora é desativada.

O LCP é utilizado para negociar opções de protocolo de enlace de dados durante a fase *ESTABLISH*. Na verdade, ele não está preocupado com as opções propriamente ditas, mas com o mecanismo de negociação. O protocolo LCP proporciona um meio para que o processo inicial faça uma proposta que será aceita ou rejeitada, total ou parcialmente, pelo processo de resposta. Ele também permite que os dois processos testem a qualidade da linha, verificando se ela é boa o suficiente para estabelecer uma conexão. Por fim, o protocolo LCP também permite que as linhas sejam desativadas quando não forem mais necessárias.

Onze tipos de quadros LCP são definidos na RFC 1661 e estão listados na Figura 3.29. Os quatro tipos *Configure-* permitem que o iniciador (I) proponha valores

de opções e que o respondedor (R) os aceite ou rejeite. Nesse último caso, o respondedor pode fazer uma proposta alternativa ou anunciar que não está absolutamente disposto a negociar certas opções. As opções que estiverem sendo negociadas e seus valores propostos fazem parte dos quadros LCP.

[arte: ver original p. 242]

[T]Tabela

Nome	Sentido	Descrição
Configure-request	I → R	Lista de opções e valores propostos
Configure-ack	I ← R	Todas as opções são aceitas
Configure-nak	I ← R	Algumas opções não são aceitas
Configure-reject	I ← R	Algumas opções não são negociáveis
Terminate-request	I → R	Solicita a desativação da linha
Terminate-ack	I ← R	0k, linha desativada
Code-reject	I ← R	Solicitação desconhecida recebida
Protocol-reject	I ← R	Protocolo desconhecido solicitado
Echo-request	I → R	Favor enviar este quadro de volta
Echo-reply	I ← R	Aqui está o quadro de volta
Discard-request	I → R	Simplesmente descartar este quadro (para fins de teste)

[F]Figura 3.29

[FL] Os tipos de quadros LCP

Os códigos *Terminate-* são utilizados para desativar uma linha quando ela não é mais necessária. Os códigos *Code-reject* e *Protocol-reject* são utilizados pelo respondedor para indicar que recebeu algo que não consegue entender. Essa situação pode significar que ocorreu um erro de transmissão não detectado, embora seja mais provável que ela signifique que o iniciador e o respondedor

estão executando versões diferentes do protocolo LCP. Os tipos *Echo-* são utilizados para testar a qualidade da linha. Por fim, utiliza-se *Discard-request* para ajudar a depuração. Se uma das extremidades estiver com problemas para obter bits do cabo, o programador poderá utilizar esse tipo para teste. Se conseguir terminar, ele será simplesmente descartado pelo receptor e não será executada qualquer outra ação, o que poderá confundir a pessoa que estiver realizando o teste.

As opções que podem ser negociadas incluem a definição do tamanho máximo da carga útil para quadros de dados, a ativação da autenticação e a escolha do protocolo a ser utilizado, a ativação do monitoramento da qualidade da linha durante a operação normal e a seleção de diversas opções de compactação de cabeçalhos.

De modo geral, há muito pouco a ser dito sobre os protocolos NCP. Cada um deles é específico para algum protocolo da camada de rede e permite que sejam feitas solicitações de configuração especificas para cada protocolo. Por exemplo, no caso do IP, a atribuição de endereços dinâmicos é a possibilidade mais importante.

[T2] 3.7 Resumo

A tarefa da camada de enlace de dados é converter o fluxo de dados sem formatação fornecido pela camada física em um fluxo de quadros a ser utilizado pela camada de rede. Diversos métodos de enquadramento são utilizados, inclusive a contagem de caracteres, a inserção de bytes e a inserção de bits. Os protocolos de enlace de dados podem oferecer recursos de controle de erros para a retransmissão de quadros danificados ou perdidos. Para evitar que um transmissor rápido sobrecarregue um receptor lento, o protocolo de enlace de dados também pode fornecer controle de fluxo. O mecanismo de janela

deslizante é bastante utilizado para integrar o controle de erros e o controle de fluxo de maneira conveniente.

Os protocolos de janela deslizante podem ser divididos em categorias pelo tamanho da janela do transmissor e pelo tamanho da janela do receptor. Quando as duas janelas são iguais a 1, o protocolo utilizado é stop-and-wait. Quando a janela do transmissor é maior que 1 (por exemplo, para impedir que o transmissor bloqueie um circuito com um longo retardo de propagação), o receptor pode ser programado para descartar todos os quadros que não o próximo quadro na seqüência, ou para armazenar no buffer os quadros fora de ordem até eles serem necessários.

Examinamos neste capítulo uma série de protocolos. O protocolo 1 se destina a um ambiente livre de erros, no qual o receptor pode manipular qualquer fluxo enviado a ele. O protocolo 2 ainda pressupõe um ambiente livre de erros, mas introduz o controle de fluxo. O protocolo 3 trata erros introduzindo números de seqüência e utilizando o algoritmo de stop-and-wait. O protocolo 4 permite a comunicação bidirecional e introduz o conceito de piggybacking. O protocolo 5 utiliza um protocolo de janela deslizante com go back n. Por fim, o protocolo 6 utiliza a retransmissão seletiva e confirmações negativas.

Os protocolos podem ser modelados com o uso de diversas técnicas para ajudar a demonstrar sua exatidão (ou a falta dela). Os modelos de máquinas de estados finitos e os modelos de redes de Petri geralmente são utilizados para essa finalidade.

Muitas redes utilizam um dos protocolos orientados a bits — SDLC, HDLC, ADCCP ou LAPB — na camada de enlace de dados. Todos esses protocolos utilizam bytes de flag para delimitar quadros, e a técnica de inserção de bits para impedir que esses bytes de flags ocorram nos dados. Todos eles também utilizam uma janela deslizante para controle de fluxo. A Internet utiliza o PPP como protocolo de

enlace de dados em linhas ponto a ponto.

[T2] Problemas

1. Um pacote de uma camada superior está dividido em 10 quadros, e cada quadro tem 80% de chances de chegar sem danos. Se o protocolo de enlace de dados não fizer qualquer controle de erros, quantas vezes em média a mensagem deverá ser enviada para que o processo inteiro seja concluído?

2. A codificação de caracteres a seguir é usada em um protocolo de enlace de dados:

A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000

Mostre a seqüência de bits transmitida (em binário) para o quadro de quatro caracteres: A B ESC FLAG quando é utilizado cada um dos métodos de enquadramento a seguir:

(a) Contagem de caracteres.

(b) Bytes de flag com inserção de bytes.

(c) Bytes de flag no início e no fim, com inserção de bits.

3. O fragmento de dados a seguir ocorre no meio de um fluxo de dados para o qual é usado o algoritmo de inserção de bytes descrito no texto: A B ESC C ESC FLAG FLAG D. Qual será a saída após a inserção?

4. Um de seus colegas, Scrooge, assinalou que é um desperdício encerrar cada quadro com um byte de flag e depois iniciar o próximo quadro com um segundo byte de flag. Um único byte de flag também poderia servir, e um byte economizado é um byte ganho. Você concorda?

5. Um string de bits, 0111101111101111110, precisa ser transmitido na camada de enlace de dados. Qual é o string realmente transmitido após a inserção de bits?

6. Quando o recurso de inserção de bits é usado, é possível que a perda, a

inserção ou a modificação de um único bit provoque um erro não detectado pelo total de verificação? Se não for possível, qual é o motivo? Se for possível, como isso é feito? O comprimento do total de verificação desempenha alguma função nesse caso?

7. Você consegue imaginar alguma circunstância em que seria preferível um protocolo de loop aberto (por exemplo, um código de Hamming) aos protocolos de feedback discutidos neste capítulo?

8. Para proporcionar maior confiabilidade que a obtida com um único bit de paridade, um esquema de codificação para detecção de erros utiliza um bit de paridade para verificar todos os bits de numeração ímpar e um segundo bit de paridade para todos os bits de numeração par. Qual é a distância de Hamming desse código?

9. As mensagens de dezesseis bits são transmitidas com o uso de um código de Hamming. Quantos bits de verificação são necessários para assegurar que o receptor poderá detectar e corrigir erros de um único bit? Mostre o padrão de bits transmitido no caso da mensagem 1101001100110101. Suponha que seja usada a paridade par no código de Hamming.

10. Um byte de 8 bits com valor binário 10101111 deve ser codificado com a utilização de um código de Hamming de paridade par. Qual é o valor binário depois da codificação?

11. Um código de Hamming de 12 bits cujo valor hexadecimal é 0xE4F chega a um receptor. Qual era o valor original em hexadecimal? Suponha que não exista mais de 1 bit com erro.

12. Uma forma de detectar erros é transmitir dados como um bloco de *n* linhas com *k* bits por linha e acrescentar bits de paridade a cada linha e a cada coluna. O canto inferior direito é um bit de paridade que verifica sua linha e sua coluna. Esse esquema detectará todos os erros simples (isolados)? E os erros duplos? E os

erros triplos?

13. Um bloco de bits com *n* linhas e *k* colunas utiliza bits de paridade horizontais e verticais para a detecção de erros. Imagine que exatamente 4 bits sejam invertidos devido a erros de transmissão. Derive uma expressão para a probabilidade de que o erro não seja detectado.

14. Qual é o resto obtido pela divisão de $x^7 + x^5 + 1$ pelo polinômio gerador $x^3 + 1$?

15. Um fluxo de bits 10011101 é transmitido com a utilização do método de CRC padrão descrito no texto. O polinômio gerador é $x^3 + 1$. Mostre o string de bit real transmitido. Suponha que o terceiro bit a partir da esquerda seja invertido durante a transmissão. Mostre que esse erro é detectado na extremidade receptora.

16. Os protocolos de enlace de dados quase sempre colocam o CRC em um final, em vez de inseri-lo no cabeçalho. Por quê?

17. Um canal tem uma taxa de bits de 4 kbps e um retardo de propagação de 20 ms. Para que faixa de variação de tamanhos de quadros a técnica stop-and-wait proporciona uma eficiência de pelo menos 50%?

18. Um tronco T1 com o comprimento de 3.000 km é utilizado para transmitir quadros de 64 bytes usando o protocolo 5. Se a velocidade de propagação for de 6 μs/km, quantos bits deverão ter os números de seqüência?

19. No protocolo 3, é possível que o transmissor inicialize o timer quando ele já estiver funcionando? Nesse caso, como isso poderia acontecer? Se não, por que é impossível?

20. Imagine que um protocolo de janela deslizante utilize tantos bits para números de seqüência, que nunca ocorra sobreposição. Que relações devem ser mantidas entre as quatro bordas da janela e o tamanho da janela, que é constante e idêntica para o transmissor e o receptor?

21. Se o procedimento *between* do protocolo 5 verificasse a condição $a \leq b \leq c$ em vez da condição $a \leq b < c$, isso teria algum efeito sobre a correção ou a eficiência do protocolo? Explique a sua resposta.

22. No protocolo 6, quando um quadro de dados chega, é feita uma verificação para confirmar se o número de seqüência é diferente do esperado, e se *no_nak* é verdadeira. Se as duas condições forem verdadeiras, será enviada uma NAK. Caso contrário, o timer auxiliar será iniciado. Imagine que a cláusula [TD]else[TN] fosse omitida. Essa alteração afetaria a correção do protocolo?

23. Imagine que o loop [TD]while[TN] de três instruções próximo ao fim do protocolo 6 fosse removido do código. Isso afetaria a correção do protocolo ou apenas o desempenho? Explique a sua resposta.

24. Suponha que o caso de erros de total de verificação fosse removido da instrução [TD]switch[TN] do protocolo 6. Como essa mudança afetaria a operação do protocolo?

25. No protocolo 6, o código de *frame_arrival* tem uma seção utilizada para NAKs. Essa seção será chamada se o quadro recebido for uma NAK e se outra condição for satisfeita. Crie uma situação em que a presença dessa outra condição seja essencial.

26. Imagine que você esteja desenvolvendo o software da camada de enlace de dados para uma linha utilizada no envio, mas não na recepção de dados. A outra extremidade da conexão utiliza o HDLC, com um número de seqüência de 3 bits e um tamanho de janela de sete quadros. Você gostaria de armazenar em buffer tantos quadros fora de seqüência quanto fosse possível, a fim de melhorar a eficiência, mas não tem permissão para modificar o software no lado do transmissor. E possível ter uma janela receptora maior que um, e ainda assim garantir que o protocolo nunca falhará? Nesse caso, qual será a maior janela que poderá ser utilizada com segurança?

27. Considere a operação do protocolo 6 sobre uma linha livre de erros de 1 Mbps. O tamanho máximo de quadro é de 1.000 bits. Novos pacotes são gerados a cada segundo. O intervalo de timeout é de 10 ms. Se o timer especial de confirmação fosse eliminado, ocorreriam timeouts desnecessários. Quantas vezes a mensagem média seria transmitida?

28. No protocolo 6, *MAX_SEQ* = $2^n - 1$. Embora essa condição seja evidentemente desejável para tornar a utilização dos bits de cabeçalho mais eficiente, não demonstramos que ela é essencial. Por exemplo, o protocolo funciona corretamente para *MAX_SEQ* = 4?

29. Quadros de 1.000 bits são enviados por um canal de 1 Mbps usando um satélite geoestacionário cujo tempo de propagação a partir da Terra é 270 ms. As confirmações são sempre transportadas por piggyback em quadros de dados. Os cabeçalhos são muito curtos. São utilizados números de seqüência de 3 bits. Qual é a utilização máxima do canal que é possível alcançar para:

(a) Stop-and-wait.

(b) Protocolo 5.

(c) Protocolo 6.

30. Calcule a fração da largura de banda desperdiçada em overhead (cabeçalhos e retransmissões) para o protocolo 6 em um canal de satélite de 50 kbps bastante carregado, contendo quadros de dados com 40 bits de cabeçalho e 3.960 bits de dados. Suponha que o tempo de propagação do sinal desde a Terra até o satélite seja 270 ms. Os quadros ACK nunca ocorrem. Os quadros NAK têm 40 bits. A taxa de erros para os quadros de dados é de 1% e para os quadros NAK é desprezível. Os números de seqüência têm 8 bits.

31. Considere um canal de satélite de 64 kbps livre de erros utilizado para enviar quadros de dados de 512 bytes em um sentido, com confirmações muito curtas voltando no outro sentido. Qual é o throughput máximo para os tamanhos de

janelas iguais a 1, 7, 15 e 127? O tempo de propagação entre a Terra e o satélite é 270 ms.

32. Um cabo com 100 Km de comprimento funciona na taxa de dados T1. A velocidade de propagação no cabo é igual a 2/3 da velocidade da luz no vácuo. Quantos bits o cabo pode conter?

33. Suponha que modelamos o protocolo 4 usando o modelo de máquina de estados finitos. Quantos estados existem para cada máquina? Quantos estados existem para o canal de comunicação? Quantos estados existem para o sistema como um todo (duas máquinas e o canal)? Ignore os erros de total de verificação.

34. Determine a seqüência de disparo para a rede de Petri da Figura 3.23 que corresponde à seqüência de estados (000), (01A), (01—), (010), (01A) da Figura 3.21. Explique com suas palavras o que a seqüência representa.

35. Com as regras de transição AC → B, B → AC, CD → E e E → CD, faça um esboço da rede de Petri descrita. A partir da rede de Petri, trace o grafo de estados finitos acessíveis a partir do estado inicial *ACD*. Qual é o conceito consagrado que essas regras de transição representam?

36. O PPP se baseia intimamente no HDLC, que utiliza a técnica de inserção de bits para evitar que bytes de flag acidentais na carga útil causem confusão. Cite pelo menos um motivo pelo qual o PPP utiliza a inserção de bytes e não a inserção de bits.

37. Qual é o overhead mínimo para o envio de um pacote IP usando o PPP? Leve em consideração apenas o overhead introduzido pelo próprio PPP, e não o overhead do cabeçalho IP.

38. O objetivo deste exercício de laboratório é implementar um mecanismo de detecção de erros usando o algoritmo de CRC padrão descrito no texto. Escreva dois programas, um gerador e um verificador. O programa gerador lê na entrada padrão uma mensagem de *n* bits que tem a forma de um string de valores 0 e 1

como uma linha de texto ASCII. A segunda linha é o polinômio de *k* bits, também em ASCII. A saída padrão é uma linha de texto ASCII com $n + k$ valores 0 e 1 que representam a mensagem a ser transmitida. Em seguida, é dada saída ao polinômio, exatamente como ele foi lido na entrada. O programa verificador lê a saída do programa gerador e transmite uma mensagem indicando se ela é correta ou não. Por fim, escreva um programa, chamado alterar, que inverta um bit na primeira linha, dependendo de seu argumento (o número do bit, considerando o bit mais à esquerda igual a 1), mas copia as duas linhas restantes de forma correta. Digitando:

[TD]gerador < arquivo | verificador [TN]

Você deverá ver que a mensagem está correta; porém, digitando

[TD]gerador < arquivo | alterar arg | verificador [TN]

você deverá obter a mensagem de erro.

39. Desenvolva um programa para simular o comportamento de uma rede de Petri. O programa deve ler as regras de transição, bem como uma lista de estados correspondentes ao momento da aceitação ou da emissão de um novo pacote por parte da @@@camada de enlace de rede. A partir do estado inicial, também lido, o programa deve escolher transições ativas ao acaso e dispará-las, verificando se um host sempre aceita dois pacotes sem que o outro host emita um novo pacote entre os dois.

[TA1]Capítulo

[TA2]4

[T1] A subcamada de controle de acesso ao meio

Como mencionamos no Capítulo 1, as redes podem ser divididas em duas categorias: as que usam conexões ponto a ponto e as que utilizam canais de difusão. Este capítulo trata das redes de difusão e de seus protocolos.

Em qualquer rede de difusão, a questão fundamental é determinar quem tem direito de usar o canal quando há uma disputa por ele. Para tornar essa questão mais clara, considere uma chamada de teleconferência, na qual seis pessoas em seis diferentes telefones estão todas conectadas entre si, de forma que cada uma pode ouvir e falar com todas as outras. É muito provável que, quando uma delas parar de falar, duas ou mais comecem a falar ao mesmo tempo, levando ao caos. Em uma reunião face a face, a confusão é evitada por meios externos. Por exemplo, em uma reunião, as pessoas levantam as mãos para pedir permissão para falar. Quando apenas um único canal está disponível, a determinação de quem deve ser o próximo a falar é muito mais difícil. Existem vários protocolos destinados a solucionar o problema, e eles formam o conteúdo deste capítulo. Na literatura, os canais de difusão às vezes são referidos como **canais de multiacesso** ou **canais de acesso aleatório.**

Os protocolos usados para determinar quem será o próximo em um canal de multiacesso pertencem a uma subcamada da camada de enlace de dados, chamada **subcamada MAC (Medium Access Control).** A subcamada MAC é especialmente importante em LANs que, em sua maioria, utilizam um canal de multiacesso como base de sua comunicação. Em contrapartida, as WANs utilizam enlaces ponto aponto, com exceção das redes de satélites. Como os canais de multiacesso têm uma relação muito íntima com as LANs, neste capítulo

trataremos das LANs em geral, bem como de algumas questões que não fazem parte estritamente da subcamada MAC.

Tecnicamente, a subcamada MAC é a parte inferior da camada de enlace de dados e, portanto, deveríamos tê-la estudado antes de analisar todos os protocolos ponto a ponto apresentados no Capítulo 3. No entanto, para a maioria das pessoas, a compreensão de protocolos que envolvem várias partes torna-se mais fácil depois que o funcionamento dos protocolos de duas partes é esclarecido. Por essa razão, nos desviamos um pouco da ordem de apresentação tradicional das camadas inferiores para as superiores.

[T2]4.1. O problema de alocação de canais

O tema central deste capítulo é definir como alocar um único canal de difusão entre usuários concorrentes. Analisaremos primeiro os esquemas estáticos e dinâmicos em geral. Em seguida, estudaremos vários algoritmos específicos.

[T3] 4.1.1. Alocação estática de canais em LANs e MANs

A maneira tradicional de alocar um único canal, tal como um tronco telefônico, entre vários usuários concorrentes é usar a FDM (Frequency Division Multiplexing). Se existem *N* usuários, a largura de banda é dividida em *N* partes do mesmo tamanho (ver Figura 2.31) e a cada usuário será atribuída uma parte. Como cada usuário tem uma banda de freqüência particular, não há interferência entre eles. Quando existe apenas um número pequeno e constante de usuários, cada um dos quais com uma carga de tráfego pesada (armazenada em buffer) — por exemplo, centrais de comutação de concessionárias — a FDM é um mecanismo de alocação simples e eficiente.

No entanto, quando o número de transmissores é grande e continuamente variável, ou quando o tráfego ocorre em rajadas, a FDM apresenta alguns

problemas. Se o espectro for dividido em *N* áreas, e menos de *N* usuários estiverem interessados em estabelecer comunicação no momento, uma grande parte do espectro será desperdiçada. Se mais de *N* usuários quiserem se comunicar, alguns deles terão o acesso negado por falta de largura de banda, mesmo que alguns dos usuários aos quais foi alocada uma banda de freqüência raramente transmitam ou recebam dados.

No entanto, mesmo supondo que o número de usuários poderia de algum modo ser mantido constante em *N*, a divisão de um único canal disponível em subcanais estáticos revela uma ineficiência inerente. O problema básico é que, quando alguns usuários ficam inativos, sua largura de banda é simplesmente perdida. Eles não estão utilizando essa largura de banda, e ninguém mais pode fazê-lo. Além disso, na maioria dos sistemas de computadores, quase todo o tráfego de dados ocorre em rajadas (são comuns relações de 1000:1 entre o tráfego de pico e o tráfego médio). Em conseqüência disso, a maioria dos canais permanecerá ociosa na maior parte do tempo.

O fraco desempenho da FDM estática pode ser visto com facilidade por um simples cálculo da teoria do enfileiramento. Vamos começar com o retardo de tempo médio, T, para um canal com capacidade C bps, e uma taxa de chegada de λ quadros/s. O comprimento de cada quadro é extraído de uma função de densidade de probabilidade exponencial com média de $1/\mu$ bits/quadro.

Com esses parâmetros, a taxa de chegadas é λ quadros/s e a taxa de serviço é μC quadros/s. Pela teoria do enfileiramento, pode-se mostrar que, para tempos de chegada e de serviço de Poisson, temos:

[Inserir equação do O.A. p. 249a]

Por exemplo, se C é 100 Mbps, o comprimento do quadro médio $1/\mu$ é 10.000 bits e a taxa de chegada de quadros λ é 5.000 quadros/s, então $T = 200$ µs.

Observe que, se ignorarmos o retardo de enfileiramento e simplesmente

perguntarmos quanto tempo é necessário para enviar um quadro de 10.000 bits em uma rede de 100 Mbps, obtermos a resposta (incorreta) de 100 μs. Esse resultado só é válido quando não há nenhuma disputa pelo canal.

Agora, vamos dividir o único canal em *N* subcanais independentes, cada um com capacidade *C*/*N* bps. A taxa média de entrada em cada um dos subcanais agora será λ/*N*. Ao recalcularmos *T*, obteremos:

[Inserir equação do O.A. p. 249b] (4-1)

O retardo médio usando FDM é *N* vezes pior do que seria se todos os quadros estivessem de alguma forma mágica organizados de maneira ordenada em uma grande fila central.

Os mesmos argumentos que se aplicam à FDM também se aplicam à TDM (Time Division Multiplexing). Para cada usuário, é alocado estaticamente o *N*-ésimo slot de tempo. Se o usuário não empregar o slot alocado, este será simplesmente desperdiçado. O mesmo é válido se dividirmos as redes fisicamente. Usando mais uma vez nosso exemplo anterior, se substituíssemos a rede de 100 Mbps por 10 redes de 10 Mbps cada uma e fizéssemos a alocação estática de cada usuário a uma delas, o retardo médio saltaria de 200 μs para 2 ms.

Como nenhum dos métodos estáticos tradicionais de alocação de canais funciona bem com um tráfego em rajadas, agora vamos tratar dos métodos dinâmicos.

[T3]4.1.2 Alocação dinâmica de canais em LANs e MANs

Antes de começarmos a descrever o primeiro dos muitos métodos de alocação de canais a serem discutidos neste capitulo, vale a pena formular cuidadosamente o problema da alocação. Existem cinco premissas fundamentais subjacentes a todo trabalho realizado nessa área, que serão descritas a seguir.

1. **Modelo da estação.** O modelo consiste em *N* **estações** independentes (computadores, telefones, comunicadores pessoais etc.), cada qual com um

programa ou usuário que gera quadros para transmissão. Algumas vezes, as estações são chamadas **terminais**. A probabilidade de um quadro ser gerado em um intervalo de duração [ver símbolo] é [ver símbolo], onde λ é uma constante (a taxa de chegada de novos quadros). Uma vez gerado um quadro, a estação é bloqueada e nada faz até que o quadro tenha sido transmitido com êxito.

2. **Premissa de canal único.** Um único canal está disponível para todas as comunicações. Todas as estações podem transmitir e receber por ele. No que se refere ao hardware, todas as estações são equivalentes, embora um software de protocolo possa atribuir prioridades a elas.

3. **Premissa de colisão.** Se dois quadros são transmitidos simultaneamente, eles se sobrepõem no tempo, e o sinal resultante é adulterado. Esse evento é denominado **colisão**. Todas as estações podem detectar colisões. Um quadro que tenha sofrido colisão terá de ser retransmitido posteriormente. Não há outros erros além dos gerados por colisões.

4a. **Tempo contínuo.** A transmissão por quadro pode começar a qualquer instante. Não há um relógio-mestre dividindo o tempo em intervalos discretos.

4b. **Tempo segmentado (slotted).** O tempo é dividido em intervalos discretos (slots). As transmissões de quadros sempre começam no início de um slot. O slot pode conter 0, 1 ou mais quadros, correspondentes a um slot ocisoso, uma transmissão bem-sucedida ou a uma colisão, respectivamente.

5a. **Detecção de portadora (carrier sense).** As estações conseguem detectar se o canal está sendo usado antes de tentarem utilizá-lo. Se for detectado que o canal está ocupado, nenhuma estação tentará usá-lo até que ele fique livre.

5b. **Não há detecção de portadora.** As estações não conseguem detectar o canal antes de tentar utilizá-lo. Elas simplesmente vão em frente e transmitem. Somente mais tarde conseguem determinar se a transmissão foi ou não bem-sucedida.

Ainda é necessário discutir essas premissas um pouco mais. A primeira diz que as estações são independentes, e que a carga é gerada a uma taxa constante. Há também a premissa de que cada estação tem apenas um programa ou usuário e, portanto, enquanto a estação estiver bloqueada, não será gerada qualquer nova carga. Os modelos mais sofisticados permitem estações multiprogramadas capazes de gerar mais carga enquanto uma estação está bloqueada, mas a análise dessas estações é muito mais complexa.

A premissa de um canal único é o núcleo do modelo. Não existem formas externas de comunicação. As estações não podem levantar as mãos para solicitar que o mestre lhes permita se comunicarem.

A premissa de colisão também é básica, embora em alguns sistemas (principalmente no espectro de dispersão) essa premissa seja abrandada, produzindo resultados surpreendentes. Além disso, algumas LANs, como as do tipo token ring, repassam entre as estações um símbolo especial, cuja posse permite ao detentor atual transmitir um quadro. Porém, nas próximas seções, vamos nos limitar ao modelo de um único canal com disputa e colisões.

São possíveis duas premissas alternativas sobre o tempo: ele é contínuo (4a) ou discreto (4b). Alguns sistemas utilizam uma delas e alguns sistemas utilizam a outra, portanto, e assim vamos descrever e analisar ambas. É óbvio que, para um determinado sistema, apenas uma delas é válida.

Da mesma forma, uma rede pode ter a detecção de portadora (5a) ou não (5b). Em geral, as LANs têm detecção de portadora. No entanto, as redes sem fios não podem usá-la de forma efetiva, porque nem toda estação pode estar dentro da faixa de rádio das outras estações. As estações em redes de detecção de portadora conectadas fisicamente podem encerrar sua transmissão de modo prematuro se detectarem que a transmissão está colidindo com outra transmissão. A detecção de colisão raramente é feita em redes sem fios, por

razões de engenharia. Observe que a palavra "portadora" (carrier) nesse sentido se refere ao sinal elétrico enviado pelo cabo, e não tem qualquer relação com concessionárias de comunicações (commom carriers) — por exemplo, as empresas de telefonia — que remontam à época do Pony Express.

[T2] 4.2 Protocolos de acesso múltiplo

Existem muitos algoritmos para alocar um canal de acesso múltiplo. Nas seções a seguir, estudaremos uma pequena amostra dos mais interessantes e apresentaremos alguns exemplos de sua utilização.

[T3] 4.2.1 ALOHA

Na década de 1970, Norman Abramson e seus colegas da Universidade do Havaí elaboraram um método novo e sofisticado para resolver o problema de alocação de canais. Desde então, seu trabalho foi ampliado por vários pesquisadores (Abramson, 1985). Embora o trabalho de Abramson, denominado sistema ALOHA, usasse a radiodifusão terrestre, a idéia básica é aplicável a qualquer sistema em que usuários descoordenados estão competindo pelo uso de um único canal compartilhado.

Descreveremos aqui duas versões do ALOHA: puro e slotted. Elas diferem quanto ao fato de o tempo estar ou não dividido em slots discretos, nos quais todos os quadros devem se ajustar. Ao contrário do slotted ALOHA, o ALOHA puro não exige a sincronização de tempo global.

[T4] ALOHA puro

A idéia básica de um sistema ALOHA é simples: permitir que os usuários transmitam sempre que tiverem dados a ser enviados. Naturalmente, haverá colisões, e os quadros que colidirem serão danificados. Porém, devido à

propriedade de feedback da difusão, um transmissor sempre consegue descobrir se seu quadro foi ou não destruído, da mesma maneira que o fazem outros usuários, bastando para isso escutar a saída do canal. Em uma LAN, esse feedback é imediato. Em um satélite, há uma demora de 270 ms antes de o transmissor saber se houve êxito na transmissão. Se não for possível por alguma razão realizar a escuta durante a transmissão, serão necessárias confirmações. Se o quadro foi destruído, o transmissor apenas espera um período de tempo aleatório e o envia novamente. O tempo de espera deve ser aleatório, pois senão os mesmos quadros continuarão a colidir repetidas vezes. Os sistemas em que vários usuários compartilham um canal comum de forma que possa gerar conflitos em geral são conhecidos como sistemas de **disputa.**

A Figura 4.1, mostra um esboço da geração de quadros em um sistema ALOHA. Os quadros foram criados com o mesmo comprimento porque o throughput dos sistemas ALOHA é maximizado quando o comprimento dos quadros é uniforme em vez de variável.

[arte: ver original p. 252]

[Dísticos]

[1]Usuário

A

B

C

D

E

[2]Tempo

[F]Figura 4.1

[FL] No ALOHA puro, os quadros são transmitidos em tempos totalmente arbitrários

Sempre que dois quadros tentarem ocupar o canal ao mesmo tempo, haverá uma colisão e ambos serão danificados. Se o primeiro bit de um novo quadro se sobrepuser apenas ao último bit de um quadro quase terminado, os dois quadros serão totalmente destruídos e terão de ser retransmitidos posteriormente. O total de verificação não consegue (e não deve) fazer distinção entre uma perda total e uma perda parcial. Quadro com erro é quadro com erro, não há distinções.

Uma questão interessante é: qual é a eficiência de um canal ALOHA? Em outras palavras, que fração de todos os quadros transmitidos escapa de colisões nessas circunstâncias tão confusas? Vamos considerar primeiro um conjunto infinito de usuários interativos em seus computadores (estações). O usuário sempre se encontra em um dentre os seguintes estados: de digitação ou espera. Inicialmente, todos os usuários estão no estado de digitação. Quando uma linha é conectada, o usuário pára de digitar e espera uma resposta. Então, a estação transmite um quadro contendo a linha e verifica o canal para saber se a transmissão foi bem-sucedida. Em caso afirmativo, o usuário vê a resposta e volta a digitar. Caso contrário, ele continua a esperar e o quadro é retransmitido continuamente até ser enviado com êxito.

O "tempo de quadro" representa o período de tempo necessário para transmitir o quadro padrão de comprimento fixo (isto é, o comprimento do quadro dividido pela taxa de bits). Nesse ponto, supomos que a população infinita de usuários gere novos quadros de acordo com uma distribuição de Poisson, com a média de N quadros por tempo de quadro. (A premissa de população infinita é necessária para garantir que N não diminuirá, à medida que os usuários forem bloqueados). Se $N > 1$, a comunidade de usuários estará gerando quadros em uma taxa superior à capacidade do canal, e praticamente todos os quadros sofrerão colisões. Para um throughput razoável, esperaríamos $0 < N < 1$.

Além dos novos quadros, as estações também geram retransmissões dos quadros que sofreram colisões anteriormente. Vamos supor ainda que a probabilidade de k tentativas de transmissão por tempo de quadro, antigas e novas combinadas, também seja uma distribuição de Poisson, com média G por tempo de quadro. Evidentemente, $G \geq N$. Em situações de carga baixa (ou seja, [ver símbolo]), ocorrerão poucas colisões e, portanto, haverá poucas retransmissões. Por conseguinte, [ver símbolo]. Em situações de carga alta, ocorrerão várias colisões e, portanto, $G > N$. Para qualquer carga, o throughput S é simplesmente a carga oferecida, G, multiplicada pela probabilidade P_0de uma transmissão ser bem-sucedida — isto é, $S = GP_0$, onde P_0 é a probabilidade de um quadro não sofrer colisão.

Um quadro não sofrerá colisão se nenhum outro for enviado dentro de um tempo de quadro a partir de seu início, como mostra a Figura 4.2. Em que condições o quadro sombreado chegará sem erros? Seja t o tempo necessário para enviar um quadro. Se qualquer outro usuário tiver gerado um quadro no intervalo entre t_0 e $t_0 + t$, o final desse quadro colidirá com o início do que quadro sombreado. Na verdade, a sorte do quadro sombreado já estava selada antes de o primeiro bit ser transmitido; porém, como em ALOHA puro uma estação não escuta o canal antes de transmitir, não há como saber se já havia outro quadro a caminho. Da mesma forma, qualquer outro quadro iniciado entre $t_0 + t$ e $t_0 + 2t$ irá colidir com o final do quadro sombreado.

[arte: ver original p. 253]

[Dísticos]

[1]Colide com o início do quadro sombreado t Colide com o final do quadro sombreado

[2] t_0 $t_0 + t$ $t_0 + 2t$ $t_0 + 3t$ Tempo

Vulnerável

[F]Figura 4.2

[FL] Período de vulnerabilidade do quadro sombreado

A probabilidade de *k* quadros serem gerados durante um determinado tempo de quadro é obtida pela distribuição de Poisson:

[Inserir equação do O.A. p. 253a] (4-2)

e, portanto, a probabilidade de zero quadros é simplesmente e^{-G}. Em um intervalo com duração de dois tempos de quadro, o número médio de quadros gerados é $2G$. A probabilidade de nenhum outro tráfego ser iniciado durante todo o período de vulnerabilidade é, portanto, indicada por $P_0 = e^{-2G}$. Usando $S = GP_0$, obtemos:

[Inserir equação do O.A. p. 253b]

A Figura 4.3 mostra a relação entre o tráfego oferecido e o throughput. O throughput máximo ocorre em $G = 0{,}5$, com $S = 1/2e$, que corresponde aproximadamente a 0,184. Em outras palavras, o melhor que podemos esperar e uma utilização de canal de 18%. Esse resultado não é muito encorajador, mas com todas as pessoas transmitindo à vontade, dificilmente poderíamos esperar uma taxa de 100% de êxito.

[arte: ver original p. 254]

[Dísticos]

[1]S (throughput por tempo de quadro)

0,40

0,30

0,20

0,10

[2] 0 0,5 1,0 1,5 2,0 3,0

G (tentativas por tempo de pacote)

[3]Slotted ALOHA: [ver símbolo]

[4]ALOHA puro: [ver símbolo]

[F]Figura 4.3

[FL] Throughput em comparação com o tráfego oferecido para sistemas ALOHA

[T4] Slotted ALOHA

Em 1972, Roberts publicou um método para duplicar a capacidade de um sistema ALOHA (Roberts, 1972). Sua proposta era dividir o tempo em intervalos discretos, com cada intervalo correspondendo a um quadro. Esse método exige que os usuários concordem em relação às fronteiras dos slots. Uma forma de alcançar a sincronização entre os usuários seria ter uma estação especial que emitisse um sinal sonoro no início de cada intervalo, como um relógio.

No método de Roberts, que passou a ser conhecido como **slotted ALOHA**, em contraste com o **ALOHA puro** de Abramson, um computador não tem permissão para transmitir sempre que um caractere de retorno de cursor é digitado. Em vez disso, é necessário esperar o início do próximo slot. Conseqüentemente, o ALOHA puro contínuo transforma-se em um sistema discreto. Como o período de vulnerabilidade está agora reduzido à metade, a probabilidade de não haver outro tráfego durante o mesmo slot do nosso quadro de teste é e^{-G}, o que nos leva a:

[Inserir equação do O.A. p. 254] (4-3)

Como podemos ver na Figura 4.3, a taxa máxima do slotted ALOHA é $G = 1$, com um throughput $S = 1/e$ ou aproximadamente 0,368, o dobro do ALOHA puro. Se o sistema estiver funcionando a uma taxa de $G = 1$, a probabilidade de um slot vazio será 0,368 (pela Equação 4-2). O melhor que podemos esperar com a utilização de um slotted ALOHA é 37% de slots vazios, 37% de sucessos e 26% de colisões. O funcionamento em valores superiores de G reduz o número de slots vazios, mas aumenta exponencialmente o número de colisões. Para ver como

ocorre esse rápido crescimento de colisões com G, considere a transmissão de um quadro de teste. A probabilidade de ele evitar uma colisão é de e^{-G}, que é a probabilidade de todos os outros usuários estarem inativos nesse slot. A probabilidade de uma colisão é, então, simplesmente $1 - e^{-G}$. A probabilidade de uma transmissão exigir exatamente k tentativas (ou seja, $k - 1$ colisões seguidas por uma transmissão bem-sucedida) é:

[Inserir equação do O.A. p. 255a]

O número esperado de transmissões, E, por cada emissão de um retorno de cursor é portanto:

[Inserir equação do O.A. p. 255b]

Como o resultado da dependência exponencial de E em relação a G, pequenos aumentos na carga do canal podem reduzir drasticamente seu desempenho.

O slotted ALOHA é importante por uma razão que a princípio talvez não seja óbvia. Ele foi criado na década de 1970, foi usado em alguns sistemas experimentais, e depois foi quase esquecido. Quando o acesso à Internet por cabo foi criado, surgiu o problema de como alocar um canal compartilhado entre vários usuários concorrentes, e o slotted ALOHA foi resgatado para salvar a situação. Com freqüência, protocolos perfeitamente válidos caem em desuso por razões políticas (por exemplo, quando alguma grande empresa deseja que todas as outras sigam seu modo de agir) mas, anos depois, alguém inteligente percebe que um protocolo descartado muito antes resolve seu problema atual. Por essa razão, estudaremos neste capítulo diversos protocolos elegantes que não são muito utilizados hoje, mas que poderiam facilmente ser empregados em aplicações futuras, desde que projetistas de redes em números suficientes tivessem consciência deles. É claro que também estudaremos muitos protocolos bastante usados atualmente.

[T3] 4.2.2 Protocolos CSMA (Carrier Sense Multiple Access)

Com o slotted ALOHA, a melhor utilização do canal que é possível conseguir é 1/*e*. Isso não surpreende porque, com as estações transmitindo à vontade, sem prestarem atenção ao que as outras estações estão fazendo, é provável que ocorram muitas colisões. Porém, em LANs as estações podem detectar o que outras estão fazendo e adaptarem seu comportamento de acordo com essa situação. Essas redes podem atingir uma utilização melhor que 1/*e*. Nesta seção, estudaremos alguns protocolos que melhoram o desempenho da rede.

Os protocolos nos quais as estações escutam uma portadora (isto é, uma transmissão) e funcionam de acordo com ela são denominados **protocolos com detecção de portadora** (carrier sense protocols). Muitos deles já foram propostos. Kleinrock e Tobagi (1975) analisaram em detalhes vários protocolos desse tipo. Mencionaremos a seguir algumas versões dos protocolos com detecção de portadora.

[T4] CSMA persistente e não persistente

O primeiro protocolo com detecção de portadora que estudaremos aqui denomina-se **CSMA (Carrier Sense Multiple Access) 1-persistente**. Quando uma estação tem dados a transmitir, ela primeiro escuta o canal para ver se mais alguém está transmitindo no momento. Se o canal estiver ocupado, a estação esperará até que ele fique ocioso. Quando detectar um canal desocupado, a estação transmitirá um quadro. Se ocorrer uma colisão, a estação esperará um intervalo de tempo aleatório e começará tudo de novo. Esse protocolo é denominado 1-persistente, porque a estação transmite com probabilidade 1 sempre que encontra o canal desocupado.

O retardo de propagação tem um efeito importante sobre o desempenho do protocolo. Há poucas chances de, logo após uma estação começar a transmitir,

outra estação fique pronta para transmitir e escutar o canal. Se o sinal da primeira estação ainda não tiver atingido à segunda, esta detectará um canal desocupado e também começará a transmitir, resultando em uma colisão. Quanto maior for o retardo de propagação, maior será a importância desse efeito e pior será o desempenho do protocolo.

Mesmo que o retardo de propagação seja zero, ainda assim haverá colisões. Se duas estações ficarem prontas durante a transmissão de uma terceira, ambas terão de esperar educamente até que a transmissão se encerre, e depois as duas começarão a transmitir ao mesmo tempo, resultando em uma colisão. Se elas não fossem tão impacientes, haveria menos colisões. Mesmo assim, esse protocolo é bem melhor que o ALOHA puro, pois ambas as estações respeitam a transmissão e desistem de interferir em um quadro de uma terceira estação. Intuitivamente, esse procedimento leva a um desempenho superior ao do ALOHA puro. O mesmo se aplica ao slotted ALOHA.

Um segundo protocolo com detecção de portadora é o **CSMA não persistente**. Nesse protocolo, é feita uma tentativa consciente de ser menos ávido que no protocolo anterior. Antes de transmitir, uma estação escuta o canal. Se ninguém mais estiver transmitindo, a estação iniciará a transmissão. No entanto, se o canal já estiver sendo utilizado, a estação não permanecerá escutando continuamente a fim de se apoderar de imediato do canal após detectar o fim da transmissão anterior. Em vez disso, a estação aguardará durante um intervalo de tempo aleatório e, em seguida, repetirá o algoritmo. Conseqüentemente, esse algoritmo leva a uma melhor utilização do canal, e a retardos maiores do que no CSMA 1-persistente.

O último protocolo é o **CSMA p-persistente**. Ele se aplica a canais segmentados (slotted channels) e funciona da forma apresentada a seguir. Quando está pronta para transmitir, a estação escuta o canal. Se ele estiver desocupado, a estação

transmitirá com uma probabilidade *p*. Com uma probabilidade *q* = 1 – *p*, haverá um adiamento até o próximo slot. Se esse slot também estiver desocupado, haverá uma transmissão ou um novo adiamento, com probabilidades *p* e *q*. Esse processo se repete até o quadro ser transmitido ou até que outra estação tenha iniciado uma transmissão. Nesse último caso, ela age como se tivesse ocorrido uma colisão (ou seja, aguarda durante um intervalo aleatório e reinicia a transmissão). Se inicialmente detectar que o canal está ocupado, a estação esperará pelo próximo slot e aplicará o algoritmo anterior. A Figura 4.4 mostra o throughput calculado em comparação com o tráfego oferecido para todos os três protocolos, bem como para o ALOHA puro e o slotted ALOHA.

[arte: ver original p. 257]

[Dísticos]

[1]S (throughput por tempo de pacote) 1,0

0,9

0,8

0,7

0,6

0,5

0,4

0,3

0,2

0,1

0

[2]0 1 2 3 4 5 6 7 8 9

G (tentativas por tempo de pacote)

[3]CSMA 0,01-persistente

CSMA não persistente

CSMA 0,1-persistente

[4]CSMA 0,5-persistente

[5]Slotted ALOHA

[6]CSMA 1-persistente

[7]ALOHA puro

[F]Figura 4.4

[FL] Comparação entre a utilização do canal e a carga de vários protocolos de acesso aleatório

[T4] CSMA com detecção de colisões

Os protocolos CSMA persistentes e não persistentes são claramente um avanço em relação ao ALOHA, pois garantem que nenhuma estação começará a transmitir quando perceber que o canal está ocupado. Outro avanço consiste no fato de as estações cancelarem suas transmissões logo que detectam uma colisão. Em outras palavras, se duas estações perceberem que o canal está desocupado e começarem a transmitir simultaneamente, ambas detectarão a colisão quase de imediato. Em vez de terminar de transmitir seus quadros que de qualquer forma já estarão irremediavelmente adulterados, elas devem interromper a transmissão de forma abrupta tão logo a colisão for detectada. A interrupção rápida dos quadros com erros economiza tempo e largura de banda. Esse protocolo, conhecido como **CSMA/CD (Carrier Sense Multiple Access with Collision Detection)**, é amplamente usado na sub-camada MAC de LANs. Em particular, ele é a base da conhecida LAN Ethernet; assim, vale a pena dedicarmos algum tempo a examiná-lo em detalhes.

O CSMA/CD e vários outros protocolos de LANs utilizam o modelo conceitual apresentado na Figura 4.5. No ponto marcado com t_0, uma estação terminou a transmissão de um quadro. Qualquer outra estação que tenha um quadro a ser

enviado pode transmiti-lo. Se duas ou mais estações decidirem transmitir simultaneamente, haverá uma colisão. As colisões podem ser detectadas verificando-se a potência e a largura do pulso do sinal recebido e comparando-o com o sinal transmitido.

Após detectar uma colisão, uma estação cancela sua transmissão, espera um intervalo de tempo aleatório e, em seguida, tenta novamente, supondo que nenhuma outra estação tenha começado a transmitir nesse ínterim. Dessa forma, o nosso modelo de CSMA/CD consistirá em períodos alternados de disputa e de transmissão, com a ocorrência de períodos de inatividade quando todas as estações estiverem em repouso (por exemplo, por falta de trabalho).

[arte: ver original p. 258]

[Dísticos]

[1]Slots em disputa

[2]t_0

[3]Quadro

Período de transmissão

[4]Período de disputa

[5]Quadro

[6]Quadro

[7]Período de inatividade

[8]Quadro

[9]Tempo

[F]Figura 4.5

[FL] O CSMA/CD pode estar em um destes três estados: disputa, transmissão ou inatividade

Agora, vamos analisar mais de perto os detalhes do algoritmo de disputa.

Suponha que duas estações comecem uma transmissão no instante exato t_0. Quanto tempo elas levarão para perceber que houve uma colisão? A resposta a essa pergunta é essencial para determinar a duração do intervalo de disputa e, portanto, o retardo e o throughput. O tempo mínimo para detecção de uma colisão é apenas o tempo que o sinal leva para se propagar de uma estação até a outra.

Com base nesse raciocínio, você poderia pensar que uma estação que não escutasse uma colisão durante um intervalo igual ao tempo de propagação em todo o cabo após ter iniciado sua transmissão poderia ter certeza de haver se apoderado do cabo. Com o termo "apoderado", queremos dizer que todas as outras estações sabiam da transmissão e não interferiram. Essa conclusão está incorreta. Considere a pior hipótese possível a seguir. Seja [ver símbolo] o tempo de propagação de um sinal entre as duas estações mais distantes. Em t_0, uma estação começa a transmitir. Em [ver símbolo], um instante antes de o sinal chegar à estação mais distante, essa estação também começa a transmitir. É claro que ela detecta a colisão quase instantaneamente e pára, mas o pequeno ruído causado pela colisão não retorna à estação original até o período de tempo [ver símbolo]. Em outras palavras, na pior das hipóteses, uma estação só poderá ter certeza de ter se apoderado do canal após transmitir durante o período [ver símbolo] sem escutar uma colisão. Por essa razão, modelaremos o intervalo de disputa como um sistema slotted ALOHA, com uma largura de slot igual a [ver símbolo]. Em um cabo coaxial de 1 Km de comprimento, [ver símbolo] μs. Para facilitar a compreensão, presumiremos que cada slot contém apenas 1 bit. É evidente que, uma vez que tenha se apoderado do canal, uma estação poderá transmitir em qualquer taxa que quiser, e não apenas a 1 bit por [ver símbolo] segundos.

É importante entender que a detecção de colisões consiste em um processo

analógico. O hardware da estação deve escutar o cabo durante a transmissão. Se o que ler for diferente do que está transmitindo, a estação saberá que está ocorrendo uma colisão. A implicação é que a codificação do sinal deve permitir que colisões sejam detectadas (por exemplo, talvez seja impossível detectar uma colisão de dois sinais de 0 volts). Por essa razão, normalmente é utilizada uma codificação especial.

Também vale a pena notar que uma estação transmissora deve monitorar continuamente o canal, em busca de rajadas de ruído que possam indicar uma colisão. Por essa razão, o CSMA/CD com um único canal é inerentemente um sistema half-duplex. É impossível uma estação transmitir e receber quadros ao mesmo tempo, porque a lógica de recepção está em uso, procurando por colisões durante cada transmissão.

Para evitar qualquer mal entendido, vale a pena notar que nenhum protocolo da subcamada MAC garante uma entrega confiável. Mesmo na ausência de colisões, é possível que o receptor não tenha copiado o quadro corretamente por várias razões (por exemplo, por falta de espaço em buffer ou devido a uma interrupção perdida).

[T3] 4.2.3 Protocolos livres de colisão

Embora as colisões não ocorram com o CSMA/CD depois que uma estação captura sem ambigüidade o canal, elas ainda podem ocorrer durante o período de disputa. Essas colisões afetam de modo adverso o desempenho do sistema, em especial quando o cabo é longo (ou seja, quando [ver símbolo] é grande) e os quadros são curtos. Além disso, o CSMA/CD não é aplicável de maneira universal. Nesta seção, examinaremos alguns protocolos que resolvem a disputa pelo canal sem a ocorrência de colisões, nem mesmo durante o período de disputa. A maioria desses protocolos não é usada atualmente em sistemas importantes mas, em um campo que muda rapidamente, a existência de alguns protocolos com excelentes propriedades disponíveis para sistemas futuros com freqüência é algo bom.

Nos protocolos que descreveremos, supomos que existem exatamente N estações, cada uma com um endereço exclusivo de 0 até $N - 1$ "conectado" a ela. O fato de que talvez algumas estações possam estar inativas durante parte do tempo não tem importância. Também supomos que o retardo de propagação é desprezível. A pergunta básica permanece: que estação terá a posse do canal após uma transmissão bem-sucedida? Continuaremos a utilizar o modelo mostrado na Figura 4.5 com seus slots discretos de disputa.

[T4] Um protocolo de mapa de bits

No nosso primeiro protocolo livre de colisão, o **método básico de mapa de bits**, cada período de disputa consiste exatamente em N slots. Se tiver um quadro para transmitir, a estação 0 enviará um bit 1 durante o slot número zero. Nenhuma outra estação poderá transmitir durante esse slot. Independente do que a estação 0 fizer, a estação 1 tem a oportunidade de transmitir um bit 1 durante o slot 1,

mas apenas se tiver um quadro na fila para ser enviado. Em geral, é possível que a estação *j* informe que tem um quadro para transmitir inserindo um bit 1 no slot *j*. Depois que todos os *N* slots tiverem passado, cada estação terá total conhecimento de quais estações desejam transmitir. Nesse ponto, elas começam a transmitir em ordem numérica (ver Figura 4.6).

[arte: imagem original da p. 259]

[Dísticos]

[1] 8 slots de disputa

0 1 2 3 4 5 6 7

1 1 1

[2] Quadros

1 3 7

[3] 8 slots de disputa

0 1 2 3 4 5 6 7

1 1 1 5

[4] 1 d

0 1 2 3 4 5 6 7

1 2

[F]Figura 4.6

[FL] O protocolo básico de mapa de bits

Como todas as estações concordam sobre quem será a próxima a transmitir, nunca haverá colisões. Após a última estação pronta ter transmitido seu quadro, um evento que todas as estações podem monitor com facilidade, inicia-se outro período de disputa de *N* bits. Se uma estação ficar pronta logo após seu slot de bits ter passado, ela não conseguirá transmitir e precisará permanecer inativa até que todas as outras estações tenham tido a chance de transmitir e o mapa de bits

tenha voltado a passar por ela. Protocolos como esse, nos quais o desejo de transmitir é difundido antes de ocorrer a transmissão real, são chamados **protocolos de reserva.**

Vamos analisar rapidamente o desempenho desse protocolo. Para facilitar, mediremos o tempo em unidades do slot de bits de disputa, com os quadros de dados consistindo em d unidades de tempo. Em condições de carga baixa, o mapa de bits será simplesmente repetido várias vezes, por falta de quadros de dados.

Considere a situação do ponto de vista de uma estação com numeração baixa, como 0 ou 1. Normalmente, quando ela fica pronta para enviar, o slot "atual" estará em algum ponto no meio do mapa de bits. Em média, a estação terá de esperar $N/2$ slots para que a varredura atual seja concluída e mais N slots completos até que se encerre a varredura seguinte, para poder começar a transmitir.

As estações que estiverem aguardando e tiverem números mais altos obterão resultados melhores. Em geral, essas estações só precisarão esperar pela metade de uma varredura ($N/2$ slots de bits) antes de iniciar a transmissão. As estações com numeração mais alta raramente precisam esperar pela próxima varredura. Como as estações de numeração baixa precisam esperar em média $1,5N$ slots e as de numeração alta precisam esperar em média $0,5N$ slots, a média para todas as estações é N slots. É fácil calcular a eficiência do canal em carga baixa. O overhead por quadro é de N bits, e o volume de dados é de d bits, o que resulta em uma eficiência igual a $d/(N + d)$.

Sob carga alta, quando todas as estações têm algo a enviar o tempo todo, o período de disputa de N bits é dividido proporcionalmente entre N quadros, produzindo um overhead de apenas 1 bit por quadro ou uma eficiência igual a $d/(d + 1)$. O retardo médio para um quadro é equivalente à soma do tempo de

espera na fila dentro da estação, mais um adicional de $N(d + 1)/2$, uma vez que ele alcança o início de sua fila interna.

[T4] Contagem regressiva binária

Um problema com o protocolo básico de mapa de bits é que o overhead é de 1 bit por estação, e portanto ele não se adapta muito bem a redes com milhares de estações. Podemos fazer melhor que isso usando endereços binários de estações. Uma estação que queira usar o canal transmite seu endereço como uma seqüência de bits binários, começando com o bit de alta ordem. Supomos que todos os endereços têm o mesmo tamanho. Os bits de cada posição de endereço das diferentes estações passam por uma operação OR booleana ao mesmo tempo. Chamaremos esse protocolo de **contagem regressiva binária**. Ele foi usado no Datakit (Fraser, 1987). Esse protocolo pressupõe implicitamente que os retardos de transmissão são desprezíveis, de forma que todas as estações detectam bits declarados quase instantaneamente.

Para evitar conflitos, precisa ser aplicada uma regra de arbitragem: assim que percebe que um bit de alta ordem que em seu endereço era 0 foi sobrescrito por um bit 1, a estação desiste. Por exemplo, se as estações 0010, 0100, 1001 e 1010 estiverem todas tentando acessar o canal, no primeiro período de um bit, as estações transmitirão 0, 0, 1 e 1, respectivamente, e esses valores passarão pela operação OR para formar um valor 1. As estações 0010 e 0100 vêem o valor 1 e sabem que uma estação de numeração mais alta está disputando o canal e, portanto, desistem da luta na rodada atual. As estações 1001 e 1010 prosseguem.

O próximo bit é 0, e as ambas as estações continuam a transmissão. O próximo bit é 1 e, portanto, a estação 1001 desiste. A vencedora é a estação 1010, pois tem o endereço mais alto. Após vencer a disputa, é provável que agora ela possa

transmitir um quadro, após o qual terá início outro ciclo de disputa. A Figura 4.7 ilustra esse protocolo. Ele tem a propriedade de dar às estações com numeração mais alta uma prioridade maior do que a prioridade concedida a estações de numeração mais baixa; isso pode ser bom ou ruim, dependendo do contexto.

[arte: imagem original da p. 261]

[Dísticos]

[1] Tempo de bit

	0 1 2 3
0 0 1 0	0 – – –
0 1 0 0	0 – – –
1 0 0 1	1 0 0 –
1 0 1 0	1 0 1 0
Resultado	1 0 1 0

[2]As estações 0010 e 0100 vêem esse valor 1 e desistem

[3]A estação 1001 vê esse valor 1 e desiste

[F]Figura 4.7

[FL] O protocolo de contagem regressiva binária. Um traço indica inatividade

Com esse método, a eficiência do canal é $d/(d + \log_2 N)$. No entanto, se o formato do quadro tiver sido corretamente escolhido, de forma que o endereço do transmissor seja o primeiro campo do quadro, mesmo esses $\log_2 N$ bits não serão desperdiçados, e a eficiência será 100%.

Mok e Ward (1979) descreveram uma variação da contagem regressiva binária usando uma interface paralela em vez de serial. Eles também sugerem o uso de números de estações virtuais de 0 em diante, incluindo o número da estação bem-sucedida que é permutado em rodízio após cada transmissão, a fim de dar prioridade mais alta a estações que ficaram inativas por um tempo

excessivamente longo. Por exemplo, se as estações *C*, *H*, *D*, *A*, *G*, *B*, *E* e *F* tiverem prioridades 7, 6, 5, 4, 3, 2, 1 e 0, respectivamente, então uma transmissão bem-sucedida por *D* incluirá essa estação no final da lista, resultando na ordem de prioridade *C*, *H*, *A*, *G*, *B*, *E*, *F* e *D*. Desse modo, *C* continuará a ser a estação virtual 7, mas a *A* passará de 4 para 5 e *D* cairá de 5 para 0. Agora, a estação *D* será a única capaz de se apoderar do canal se nenhuma outra estação o quiser.

A contagem regressiva binária é um exemplo de protocolo simples, elegante e eficiente que está esperando o momento de ser redescoberto. Esperamos que ele algum dia encontre um novo período de sucesso.

[T3] 4.2.4 Protocolos de disputa limitada

Já vimos até agora duas estratégias básicas para a aquisição de canais em uma rede conectada por cabos: métodos de disputa, como no CSMA, e métodos livres de colisão. Cada estratégia é classificada de acordo com seu desempenho em relação a duas medidas importantes, o retardo em carga baixa e a eficiência de canal em carga alta. Em condições de carga leve, a disputa (ou seja, o ALOHA puro ou o slotted ALOHA) é preferível, em virtude de seu baixo índice de retardo. À medida que a carga aumenta, a disputa torna-se cada vez menos interessante, pois o overhead associado à arbitragem do canal torna-se maior. O oposto também é verdadeiro em relação aos protocolos livres de colisão. Em carga baixa, eles têm um alto índice de retardo, mas à medida que a carga aumenta, a eficiência do canal melhora em vez de piorar, como ocorre nos protocolos de disputa.

Obviamente, seria bom se pudéssemos combinar as melhores propriedades dos protocolos de disputa e dos protocolos livres de colisão. Dessa forma, criaríamos um novo protocolo que usaria não só a disputa em cargas baixas, para proporcionar um baixo índice de retardo, como também a técnica livre de colisão

em carga alta, para oferecer uma boa eficiência de canal. Esses protocolos, que chamaremos **protocolos de disputa limitada** existem de fato, e concluirão o nosso estudo sobre redes com detecção de portadora.

Até agora, os únicos protocolos de disputa que estudamos são simétricos, isto é, cada estação tenta acessar o canal com a mesma probabilidade p, com todas as estações usando o mesmo p. É interessante observar que o desempenho geral do sistema às vezes pode ser melhorado com o uso de um protocolo que atribua probabilidades distintas a diferentes estações.

Antes de examinarmos os protocolos assimétricos, faremos uma pequena revisão do desempenho no caso simétrico. Suponha que k estações estejam disputando o acesso a um canal. Cada uma tem a probabilidade p de transmitir durante cada slot. A probabilidade de alguma estação acessar o canal com sucesso durante determinado slot é $kp(1 - p)^{k-1}$. Para encontrar o valor ótimo de p, diferenciamos em relação a p, definimos o resultado como zero e resolvemos a equação para p. Ao fazer isso, descobrimos que o melhor valor de p é $1/k$. Ao substituirmos $p = 1/k$, obtemos:

[Inserir equação do O.A. p. 262]

(a) [sucesso com p ótimo] (4-4)

Essa probabilidade está representada na Figura 4.8. Para um pequeno número de estações, as chances de sucesso são boas, mas tão logo o número de estações alcança até mesmo cinco, a probabilidade cai até um valor próximo de seu valor assintótico, $1/e$.

Na Figura 4.8, é óbvio que a probabilidade de alguma estação adquirir o canal só pode ser aumentada diminuindo-se o volume de competição. Os protocolos de disputa limitada fazem exatamente isso. Primeiro, eles dividem as estações em grupos (não necessariamente disjuntos). Apenas os membros do grupo 0 podem disputar o slot 0. Se um deles obtiver êxito, adquire o canal e transmite seu

quadro. Se um slot permanecer inativo ou se ocorrer uma colisão, os membros do grupo 1 disputarão o slot 1 etc. Fazendo-se uma divisão apropriada das estações em grupos, o volume de disputa por cada slot pode ser reduzido, e assim a operação de cada slot ficará próxima à extremidade esquerda da Figura 4.8.

O truque é a maneira de atribuir estações a slots. Antes de analisarmos o caso geral, vamos considerar algumas situações especiais. Em um extremo, cada grupo tem apenas um membro. Essa atribuição garante que nunca ocorrerão colisões, pois existirá no máximo uma estação disputando qualquer slot dado. Já vimos esse tipo de protocolo antes (por exemplo, a contagem regressiva binária).

A próxima situação especial é atribuir duas estações por grupo. A probabilidade de ambas tentarem transmitir durante um slot é p^2 que, para p pequeno, é desprezível. À medida que mais e mais estações são atribuídas ao mesmo slot, a probabilidade de colisão aumenta, mas diminui a extensão da varredura de mapa de bits necessária para que todas tenham uma chance. A situação limite consiste em um único grupo que contém todas as estações (ou seja, o slotted ALOHA). O que precisamos é de uma forma de atribuir dinamicamente estações a slots, com várias estações por slot quando a carga for baixa, e poucas estações (ou apenas uma) por slot quando a carga for alta.

[arte: imagem original da p. 263]

[Dísticos]

[1]Probabilidade de sucesso

[2]1,0

0,8

0,6

0,4

0,2

0,0

[3]0 5 10 15 20 25

[4]Número de estações prontas

[F]Figura 4.8

[FL] Probabilidade de aquisição de um canal de disputa simétrico

[T4] O protocolo adaptativo de percurso em árvore

Uma maneira particularmente simples de fazer as atribuições necessárias consiste em usar o algoritmo desenvolvido pelo exército norte-americano para testar a incidência de sífilis em soldados durante a Segunda Guerra Mundial (Dorfman, 1943). Em resumo, o exército extraiu uma amostra de sangue de *N* soldados. Uma parte de cada amostra foi colocada em um único tubo de teste. Então, verificou-se se havia anticorpos nessa amostra misturada. Se nenhum anticorpo fosse encontrado, todos os soldados do grupo eram considerados saudáveis. Se houvesse anticorpos, duas novas amostras misturadas eram preparadas, uma dos soldados numerados de 1 a *N*/2 e outra com o sangue dos demais soldados. O processo era repetido recursivamente até que os soldados infectados fossem identificados.

Para a versão computacional desse algoritmo (Capetanakis, 1979), é conveniente imaginar as estações como as folhas de uma árvore binária, conforme ilustra a Figura 4.9. No primeiro slot de disputa que segue uma transmissão de quadro bem-sucedida, o slot 0, todas as estações têm permissão para tentar acessar o canal. Se uma delas conseguir, muito bem. Se ocorrer uma colisão, durante o slot 1, apenas as estações que estiverem sob o nó 2 da árvore poderão disputar o canal. Se uma delas se apoderar do canal, o slot seguinte ao quadro ficará reservado para as estações do nó 3. Por outro lado, se duas ou mais estações no nó 2 quiserem transmitir, ocorrerá uma colisão durante o slot 1 e, nesse caso, será a vez do nó 4 durante o slot 2.

[arte: imagem original da p. 264]

[Dísticos]

[1] Estações

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 4.9

[FL] A árvore para oito estações

Basicamente, se ocorrer uma colisão durante o slot 0, toda a árvore será pesquisada, primeiro na profundidade, a fim de localizar todas as estações prontas para transmissão. Cada slot de bits é associado a algum nó específico da árvore. Se ocorrer uma colisão, a pesquisa continuará recursivamente com os filhos localizados à esquerda e à direita desse nó. Se um slot de bits estiver inativo ou se houver apenas uma estação transmitindo nesse slot, a pesquisa de seu nó poderá ser encerrada, pois todas as estações prontas terão sido localizadas. (Se houvesse mais de uma, teria ocorrido uma colisão.)

Quando a carga do sistema está muito pesada, quase não vale a pena o esforço de dedicar o slot 0 ao nó 1, pois esse procedimento só faz sentido na eventualidade improvável de que exatamente uma estação tenha um quadro a ser transmitido. Assim, alguém poderia argumentar que os nós 2 e 3 também deveriam ser ignorados, pela mesma razão. Em termos mais gerais, em que nível da árvore a pesquisa deve ter início? É claro que, quanto maior for a carga, mais baixo na árvore deve estar o ponto de início da pesquisa. Por ora, vamos supor que cada estação tem uma boa estimativa do número q de estações prontas, por exemplo, com base no monitoramento do tráfego mais recente.

Para prosseguir, vamos numerar os níveis da árvore a partir do topo, com o nó 1

da Figura 4.9 no nível 0, os nós 2 e 3 no nível 1 etc. Observe que cada nó do nível i tem uma fração 2^{-i} das estações que se encontram abaixo dele. Se as q estações prontas estiverem uniformemente distribuídas, o número esperado dessas estações abaixo de um nó específico do nível i será apenas $2^{-i}q$. Intuitivamente, seria de esperar que o nível ideal para iniciar a pesquisar na árvore fosse aquele no qual o número médio de estações em disputa por slot fosse igual 1, isto é, o nível em que $2^{-i}q = 1$. Resolvendo essa equação, descobrimos que $i = \log_2 q$.

O algoritmo básico já foi amplamente aperfeiçoado, e esses avanços são abordados em detalhes por Bertsekas e Gallager (1992). Por exemplo, considere a hipótese em que as estações *G* e *H* são as únicas que estão esperando para transmitir. No nó 1, ocorrerá uma colisão e, assim 2 será descoberto como nó inativo. É inútil testar o nó 3, pois é certo que haverá colisão (sabemos que duas ou mais estações abaixo de 1 estão prontas e que nenhuma delas está abaixo de 2; portanto, todas devem estar abaixo de 3). A sondagem do nó 3 pode ser ignorada, e o nó 6 será testado em seguida. Quando essa sondagem também não produzir qualquer resultado, 7 poderá ser ignorado, e o nó *G* poderá ser testado em seguida.

[T3] 4.2.5 Protocolos WDMA

Outro método para a alocação de canais consiste em dividir o canal em subcanais utilizando FDM, TDM ou ambas, e alocá-los dinamicamente, de acordo com as necessidades. Esquemas como esse são usados normalmente em LANs de fibra óptica para permitir que conversações distintas utilizem comprimentos de onda (ou seja, freqüências) diferentes ao mesmo tempo. Nesta seção, examinaremos esse protocolo (Humblet *et al.*, 1992).

Uma maneira simples de construir uma LAN totalmente óptica consiste em usar

um acoplador de estrela passiva (ver Figura 2.10). Na verdade, duas fibras de cada estação são fundidas em um cilindro de vidro. Uma fibra é destinada à saída para o cilindro, e outra se destina à entrada a partir do cilindro. Uma saída de luz de qualquer estação ilumina o cilindro e pode ser detectada por todas as outras estações. As estrelas passivas podem tratar centenas de estações.

Para permitir várias transmissões ao mesmo tempo, o espectro é dividido em canais (bandas de comprimento de onda), como mostra a Figura 2.31. Nesse protocolo, denominado **WDMA (Wavelength Division Multiple Access — acesso múltiplo por divisão de comprimento de onda)**, são atribuídos dois canais a cada estação. Um canal estreito é fornecido como canal de controle para transmitir sinais à estação, e um canal largo é oferecido para que a estação possa transmitir quadros de dados.

[arte: imagem original da p. 265]

[Dísticos]

[1] m slots de tempo para controle

Estação

A S S

[2]O canal de controle de A é usado por outras estações para entrar em contato com A

[3]n slots de tempo para dados e mais 1 para status

Canal de controle de B

B S Canal de dados de B

[4]Usado por B para transmitir dados

[5]Canal de controle de C λ

C S Canal de dados de C

[6]Canal de controle de D

D S Canal de dados de D

Tempo

[F]Figura 4.10

[FL] Acesso múltiplo por divisão de comprimento de onda (WDMA — Wavelength Division Multiple Access)

Cada canal é dividido em grupos de slots de tempo, como mostra a Figura 4.10. Vamos chamar de *m* o número de slots do canal de controle, e de $n + 1$ o número de slots do canal de dados, onde *n* desses slots são destinados a dados e o último é utilizado por uma estação para informar seu status (principalmente, sobre quais slots dos dois canais estão livres). Nos dois canais, a seqüência de slots é repetida indefinidamente, e o slot 0 é marcado de maneira especial para que retardatários possam detectá-lo. Todos os canais são sincronizados por um único relógio global.

O protocolo aceita três classes de tráfego: (1) tráfego orientado a conexões com taxa de dados constante, como vídeo não compactado, (2) tráfego orientado a conexões com taxa de dados variável, como transferência de arquivos e (3) tráfego de datagramas, como pacotes UDP. Nos dois protocolos orientados a conexões a idéia é que, para *A* se comunicar com *B*, primeiro ela precisa inserir um quadro CONNECTION REQUEST (SOLICITAÇÃO DE CONEXÃO) em um slot livre do canal de controle de *B*. Se *B* aceitar, a comunicação poderá ser estabelecida no canal de dados de *A*.

Cada estação tem dois transmissores e dois receptores, como mostramos a seguir.

1. Um receptor de comprimento de onda fixo para ouvir seu próprio canal de controle.

2. Um transmissor ajustável para transmissão nos canais de controle de outras estações.

3. Um transmissor de comprimento de onda fixo para transmissão de quadros de dados.
4. Um receptor ajustável para selecionar um transmissor de dados para escuta.

Em outras palavras, cada estação detecta se há solicitações recebidas em seu próprio canal de controle, mas precisa ajustar-se ao comprimento de onda do transmissor para receber os dados. O ajuste do comprimento de onda é executado por um interferômetro de Fabry-Perot ou Mach-Zehnder, que filtra todos os comprimentos de onda, exceto a banda de comprimento de onda desejada.

Agora, vamos analisar como a estação *A* configura um canal de comunicação da classe 2 com a estação *B* para, digamos, transferência de arquivos. Em primeiro lugar, *A* ajusta seu receptor de dados ao canal de dados de *B* e aguarda o slot de status. Esse slot informa quais slots de controle estão atribuídos e quais estão livres no momento. Por exemplo, vemos na Figura 4.10 que, dos oito slots de controle de *B*, os slots 0, 4 e 5 estão livres. Os restantes estão ocupados (como indica a figura).

A estação *A* escolhe um dos slots de controle livres (digamos, o slot 4) e insere sua mensagem CONNECTION REQUEST desse slot. Como monitora constantemente seu canal de controle, *B* vê a solicitação e, em resposta, atribui o slot 4 a *A*. Essa atribuição é informada no slot de status do canal de dados de *B*. Quando *A* vê a informação, sabe que tem uma conexão unidirecional. Se *A* solicitasse uma conexão bidirecional, *B* repetiria agora o mesmo algoritmo com *A*.

É possível que, ao mesmo tempo em que *A* tentou se apoderar do controle do slot 4 de *B*, *C* tenha feito o mesmo. Nenhuma das duas estações terá êxito, e ambas detectarão a falha monitorando o slot de status no canal de controle de *B*. Elas agora aguardarão um período de tempo aleatório e tentarão novamente mais tarde.

Nesse ponto, cada parte tem um caminho livre de conflitos para enviar breves mensagens de controle à outra parte. Para executar a transferência de arquivos, *A* envia a *B* uma mensagem de controle dizendo, por exemplo: "Por favor, observe minha próxima saída de dados no slot 3. Há um quadro de dados para você nesse local." Quando *B* recebe a mensagem de controle, ajusta seu receptor ao canal de saída de *A* para ler o quadro de dados. Dependendo do protocolo da camada mais alta, *B* pode usar o mesmo mecanismo para enviar uma confirmação, se desejar.

Observe que surgirá um problema se as estações *A* e *C* tiverem conexões com *B* e cada uma delas orientar *B* a observar o slot 3. A estação *B* escolherá uma dessas solicitações ao acaso, e a outra transmissão será perdida.

Em um tráfego de taxa constante, é utilizada uma variação desse protocolo. Quando solicita uma conexão, a estação A envia simultaneamente a seguinte pergunta: "Posso lhe enviar um quadro em cada ocorrência do slot 3?" Se *B* for capaz de aceitar (ou seja, não tiver qualquer compromisso anterior em relação ao slot 3), será estabelecida uma conexão com largura de banda garantida. Do contrário, *A* poderá tentar novamente com uma outra proposta, dependendo dos slots de saída que ela tiver disponíveis.

O tráfego da classe 3 (datagramas) também utiliza outra variação. Em vez de inserir uma mensagem CONNECTION REQUEST no slot de controle que acabou de encontrar (4), ele insere uma mensagem DATA FOR YOU IN SLOT 3 (DADOS PARA VOCÊ NO SLOT 3). Se *B* estiver livre durante o próximo slot de dados 3, a transmissão será concluída com sucesso. Do contrário, o quadro de dados se perderá. Dessa maneira não será mais necessária nenhuma conexão.

É possível que existam diversas variantes em todo o protocolo. Por exemplo, em vez de atribuir a cada estação seu próprio canal de controle, um único canal de controle pode ser compartilhado por todas as estações. A cada estação é atribuído um bloco de slots de cada grupo, multiplexando efetivamente vários

canais virtuais em um único canal físico.

Também é possível utilizar um único transmissor e um único receptor ajustáveis por estação, dividindo-se o canal de cada estação em *m* slots de controle, seguidos por *n* + 1 slots de dados. A desvantagem nesse caso é que os transmissores têm de esperar mais tempo para capturar um slot de controle, e os quadros de dados consecutivos ficarão mais afastados, porque algumas informações de controle estarão a caminho.

Foram propostos e implementados vários outros protocolos WDMA, diferentes em diversos detalhes. Alguns só têm um canal de controle, outros têm vários canais de controle. Alguns levam em conta o retardo de propagação, enquanto outros não o fazem. Alguns tornam o tempo de ajuste uma parte explícita do modelo, e outros o ignoram. Os protocolos também diferem em termos de complexidade de processamento, throughput e escalabilidade. Quando está sendo usado um grande número de freqüências, o sistema costuma ser chamado **DWDM (Dense Wavelength Division Multiplexing — multiplexação por divisão de comprimento de onda denso)**. Para obter mais informações, consulte (Bogineni *et al.*, 1993; Chen, 1994; Goralski, 2001; Kartalopoulos, 1999; e Levine e Akyildiz, 1995).

[T3] 4.2.6 Protocolos de LANs sem fios

À medida que cresce o número de dispositivos móveis de comunicação e computação, também aumenta a demanda para conectá-los ao mundo exterior. Mesmo os primeiros telefones móveis tinham a capacidade de se conectar a outros telefones. Os primeiros computadores portáteis não tinham esse recurso mas, logo depois, os modems se tornaram comuns em notebooks. Para estabelecerem comunicação, esses computadores tinham de ser conectados a uma tomada de telefone. A necessidade de uma conexão física com a rede fixa significava que os computadores eram portáteis, mas não móveis.

Para alcançar a verdadeira mobilidade, os notebooks precisam usar sinais de rádio (ou infravermelho) para comunicação. Dessa forma, os usuários dedicados podem ler e enviar mensagens de correio eletrônico enquanto estão dirigindo ou velejando. Um sistema de notebooks que se comunicam por rádio pode ser considerado uma LAN sem fio, como discutimos na Seção 1.5.4. Essas LANs têm propriedades um pouco diferentes daquelas que caracterizam as LANs convencionais e exigem o uso de protocolos especiais da subcamada MAC. Nesta seção, analisaremos alguns desses protocolos. Você pode encontrar mais informações sobre as LANs sem fios em (Geier, 2002; e O'Hara e Petrick, 1999).
Uma configuração comum para uma LAN sem fio é um edifício comercial com estações base (também chamadas pontos de acesso) estrategicamente posicionadas no edifício. Todas as estações base são interconectadas com o uso de cobre ou fibra. Se a potência de transmissão das estações base e dos notebooks for ajustada para um alcance de 3 ou 4 metros, cada sala se tornará uma única célula, e o edifício inteiro passará a ser um grande sistema celular, assim como os sistemas telefônicos celulares tradicionais que estudamos no Capítulo 2. Ao contrário dos sistemas telefônicos celulares, cada célula só tem um canal, que cobre toda a largura de banda disponível e todas as estações em sua célula. Em geral, sua largura de banda é de 11 a 54 Mbps.
Nos exemplos a seguir, por simplicidade, iremos supor que todos os transmissores de rádio têm um alcance fixo. Quando um receptor estiver dentro do alcance de dois transmissores ativos, em geral o sinal resultante apresentará interferência e será inútil; em outras palavras, não consideremos mais os sistemas do tipo CDMA nessa discussão. É importante perceber que, em algumas LANs sem fios, nem todas as estações estão dentro do alcance de alguma outra estação, o que gera diversas complicações. Além disso, para LANs sem fios internas, a presença de paredes entre as estações pode produzir um impacto

decisivo sobre o alcance efetivo de cada estação.

Um método simples de usar uma LAN sem fio talvez seja experimentar o CSMA: apenas ouvir outras transmissões e só transmitir se ninguém mais estiver fazendo isso. O problema é que, na verdade, esse protocolo não é apropriado, pois o que importa é a interferência no receptor e não no transmissor. Para observar a natureza do problema, considere a Figura 4.11, onde são representadas quatro estações sem fios. Para nossos objetivos, não importa quais delas são estações base e quais são notebooks. O alcance de rádio é definido de forma que *A* e *B* fiquem dentro do alcance uma da outra, havendo possibilidade de interferência entre elas. *C* também pode interferir com *B* e *D*, mas não com *A*.

[arte: imagem original da p. 268]

[Dísticos]

[1] A B C D

Alcance de rádio

(a)

[2] A B C D

(b)

[F]Figura 4.11

[FL] Uma LAN sem fio. (a) *A* está transmitindo. (b) *B* está transmitindo

Considere primeiro o que acontece quando *A* está transmitindo para *B*, como mostra a Figura 4.11(a). Se detectar o meio físico, *C* não ouvirá *A*, pois essa estação está fora do alcance e, portanto, concluirá incorretamente que pode fazer a transmissão para *B*. Se não começar a transmitir, *C* interferirá com *B*, removendo o quadro de *A*. O problema de uma estação não conseguir detectar uma provável concorrente pelo meio físico, porque a estação concorrente está muito longe, é denominado **problema da estação oculta**.

Agora, vamos considerar a situação inversa: *B* está transmitindo para *A*, como mostra a Figura 4.11(b). Se detectar o meio físico, *C* ouvirá uma transmissão em andamento e concluirá incorretamente que não pode transmitir para *D* quando, na verdade, essa transmissão só geraria uma recepção de má qualidade na zona entre *B* e *C*, onde nenhum dos receptores desejados está localizado. Essa situação é chamada **problema da estação exposta.**

O problema é que antes de iniciar uma transmissão, a estação realmente deseja saber se há ou não atividade no receptor. O CSMA apenas informa a ela se há ou não atividade na estação que detecta a portadora. Com um fio, todos os sinais se propagam para todas as estações e, portanto, somente uma transmissão pode ocorrer de cada vez em qualquer parte do sistema. Em um sistema baseado em ondas de rádio de pequeno alcance, várias transmissões podem ocorrer simultaneamente, se todas tiverem destinos diferentes e esses destinos estiverem fora do alcance uns dos outros.

Outra maneira de refletir sobre esse problema é imaginar um edifício comercial em que todos os funcionários têm notebooks sem fios. Suponha que Linda queira enviar uma mensagem a Leonardo. O computador de Linda detecta o ambiente local e, ao detectar a ausência de atividade, inicia a transmissão. No entanto, talvez ainda ocorra uma colisão no escritório de Leonardo, pois é possível que uma terceira pessoa também esteja enviando alguns dados para ele de um local tão distante de Linda que ela não consegue detectá-lo.

[T4] MACA e MACAW

Um protocolo antigo criado para LANs sem fios é o **MACA (Multiple Access with Collision Avoidance — acesso múltiplo com abstenção de colisão)** (Karn, 1990). A idéia básica consiste em fazer com que o transmissor estimule o receptor a liberar um quadro curto como saída, para que as estações vizinhas possam

detectar essa transmissão e evitar transmitir enquanto o quadro de dados (grande) estiver sendo recebido. A Figura 4.12 mostra o protocolo MACA.

Vamos analisar agora como *A* envia um quadro para *B*. *A* inicia a transmissão enviando um quadro **RTS (Request to Send)** para *B*, como mostra a Figura 4.12(a). Esse quadro curto (30 bytes) contém o comprimento do quadro de dados que eventualmente será enviado em seguida. Depois disso, *B* responde com um quadro **CTS (Clear to Send)**, como mostra a Figura 4.12(b). O quadro CTS contém o tamanho dos dados (copiado do quadro RTS). Após o recebimento do quadro CTS, *A* inicia a transmissão.

Agora vamos ver como reagem as estações que não conseguem ouvir esses quadros. Qualquer estação que esteja ouvindo o quadro RTS está próxima a *A* e deve permanecer inativa por tempo suficiente para que o CTS seja transmitido de volta para *A*, sem conflito. Qualquer estação que esteja ouvindo o CTS está próxima a *B* e deve permanecer inativa durante a transmissão de dados que está a caminho, cujo tamanho pode ser verificado pelo exame do quadro CTS.

[arte: imagem original da p. 270]

[Dísticos]

[1] Alcance do transmissor de A

[2] Alcance do transmissor de B

[3] C A RTS B D

E

(a)

[4] C A CTS B D

E

(b)

[F]Figura 4.12

[FL] O protocolo MACA. (a) *A* está enviando um quadro RTS para *B*. (b) *B* está

respondendo com um quadro CTS para *A*

Na Figura 4.12, *C* está dentro do alcance de *A*, mas não no alcance de *B*. Portanto, essa estação pode detectar a RTS de *A*, mas não a CTS de *B*. Desde que não interfira com a CTS, a estação é livre para transmitir enquanto o quadro de dados está sendo enviado. Em contraste, *D* está dentro do alcance de *B*, mas não de *A*. Ela não detecta a RTS, mas sim a CTS. Ao detectar a CTS, ela recebe a indicação de que está perto de uma estação que está prestes a receber um quadro e, portanto, adia a transmissão até o momento em que a transmissão desse quadro deve ter sido concluída. A estação *E* detecta as duas mensagens de controle e, como *D*, deve permanecer inativa até que a transmissão do quadro de dados seja concluída.

Apesar dessas precauções, ainda pode haver colisões. Por exemplo, *B* e *C* poderiam enviar quadros RTS para *A* ao mesmo tempo. Haverá uma colisão entre esses quadros e eles se perderão. No caso de uma colisão, um transmissor que não obtiver êxito (ou seja, o que não detectar uma CTS no intervalo de tempo esperado) aguardará durante um intervalo aleatório e tentará novamente mais tarde. O algoritmo utilizado é o recuo binário exponencial, que estudaremos quando começarmos a analisar o padrão Ethernet.

Com base em estudos de simulação do MACA, Bharghavan *et al.* (1994) otimizaram o MACA para melhorar seu desempenho e deram ao novo protocolo o nome **MACAW (MACA for Wireless).** Logo no início, eles observaram que sem as confirmações da camada de enlace de dados, os quadros perdidos não eram retransmitidos até que a camada de transporte percebesse sua ausência, bem mais tarde. Eles resolveram esse problema introduzindo um quadro ACK após cada quadro de dados bem-sucedido. Os pesquisadores também observaram que o CSMA tinha alguma utilidade — principalmente para impedir uma estação de

transmitir uma RTS ao mesmo tempo que outra estação vizinha também estiver transmitindo para o mesmo destino. Portanto, a detecção de portadora passou a ser utilizada. Além disso, eles decidiram utilizar o algoritmo de recuo individualmente para cada fluxo de dados (par origem-destino), e não para cada estação. Essa mudança melhorou a precisão do protocolo. Por fim, foi incluído um mecanismo para que as estações trocassem informações sobre congestionamento, e também uma forma de fazer o algoritmo de recuo reagir de modo menos violento a problemas temporários, o que melhorou o desempenho do sistema.

[T2] 4.3 Ethernet

Agora concluímos nossa abordagem abstrata geral sobre protocolos de alocação de canais e, portanto, é hora de analisarmos como esses princípios se aplicam a sistemas reais, em especial às LANs. Como explicamos na Seção 1.5.3, o IEEE padronizou várias redes locais e metropolitanas com o nome IEEE 802. Alguns desses padrões sobreviveram, mas muitos não, como vimos na Figura 1.38. Algumas pessoas que acreditam em reencarnação crêem que Charles Darwin retornou como membro da associação de padrões do IEEE com a finalidade de eliminar os menos capazes. Os mais importantes entre os sobreviventes são o padrão 802.3 (Ethernet) e o 802.11 (LAN sem fio). É cedo demais para falar do 802.15 (Bluetooth) e do 802.16 (MAN sem fio); consulte a quinta edição deste livro para saber se eles sobreviveram. O 802.3 e o 802.11 têm camadas físicas diferentes e subcamadas MAC diferentes, mas convergem para a mesma subcamada de controle de enlace lógico (definida no padrão 802.2), e portanto têm a mesma interface para a camada de rede.

Introduzimos a Ethernet na Seção 1.5.3 e não repetiremos esse assunto aqui. em vez disso, vamos nos concentrar nos detalhes técnicos da Ethernet, nos

protocolos e nos desenvolvimentos recentes em Ethernet de alta velocidade (gigabit). Tendo em vista que o padrão Ethernet e o IEEE 802.3 são idênticos, exceto por duas diferenças secundárias que discutiremos em breve, muitas pessoas utilizam os termos "Ethernet" e "IEEE 802.3" de modo intercambiável, e nós também o faremos. Para obter mais informações sobre a Ethernet, consulte (Breyer e Riley, 1999; Seifert, 1998; e Spurgeon, 2000).

[T3] 4.3.1 Cabeamento Ethernet

Como o nome "Ethernet" se refere ao cabo (o éter), vamos iniciar nossa discussão por esse ponto. Quatro tipos de cabeamento são usados comumente, como mostra a Figura 4.13.

[arte: ver original p. 271]

[Tabela]

Nome	Cabo	Máximo de seg.	Nós/seg.	Vantagens
10Base5	Coaxial grosso	500 m	100	Cabo original; agora obsoleto
10Base2	Coaxial fino	185 m	30	Sem necessidade de hubs
10Base-T	Par trançado	100 m	1024	Sistema mais econômico
10Base-F	Fibra óptica	2000 m	1024	Melhor entre edifícios

[F]Figura 4.13

[FL] Os tipos mais comuns de cabeamento Ethernet

Do ponto de vista histórico, o cabeamento **10Base5**, popularmente chamado **Ethernet grosso**, surgiu primeiro. Ele se parece com uma mangueira de jardim amarela, com marcações a cada 2,5 m, para mostrar onde devem ser encaixados

os conectores de pressão (ou derivações). (O padrão 802.3 não *exige* que o cabo seja amarelo, mas assim o *sugere*.) Em geral, as conexões são realizadas com **conectores de pressão (vampire taps)**, nos quais um pino é *muito* cuidadosamente inserido até a metade na parte central do cabo coaxial. A notação 10Base5 significa que ele opera a 10 Mbps, utiliza a sinalização de banda básica e pode aceitar segmentos de até 500 metros. O primeiro número é a velocidade em Mbps. Em seguida, temos a palavra "Base" (ou, algumas vezes, "BASE") para indicar a transmissão de banda básica. No início, existia uma variante de banda larga, a 10Broad36, mas ela nunca teve sucesso no mercado e, portanto, desapareceu. Por fim, se o meio for coaxial, seu comprimento será arredondado para unidades de 100 metros depois de "Base".

Historicamente, o segundo tipo de cabo foi o **10Base2**, ou **Ethernet fino** que, em contraste com os cabos do tipo mangueira de jardim, usados pelo Ethernet grosso, é bem mais flexível. As conexões para esse cabo são feitas com o uso de conectores BNC padrão para formar junções em T, em vez de usar derivações. Os conectores BNC são mais fáceis de usar e mais confiáveis. O Ethernet fino é muito mais econômico e mais fácil de instalar, mas só pode ter 185 metros por segmento, cada um dos quais pode manipular apenas 30 máquinas.

A detecção de cabos partidos, comprimento excessivo, conectores defeituosos ou conectores frouxos pode representar um grande problema nos dois meios. Por essa razão, foram desenvolvidas técnicas para detectar esses problemas. Basicamente, é injetado no cabo um pulso de forma conhecida. Se o pulso atingir um obstáculo ou o fim do cabo, um eco será gerado e enviado de volta. Cronometrando cuidadosamente o intervalo entre o envio do pulso e a recepção do eco, é possível localizar a origem do eco. Essa técnica é denominada **@@@refletometria por domínio de tempo**.

Os problemas associados à localização de cabos partidos levaram os sistemas a

utilizarem outro tipo de padrão de fiação, no qual todas as estações têm um cabo conectado a um **hub** central; nesse hub, todas as estações estão conectadas eletricamente (como se estivessem soldadas juntas). Em geral, esses fios são pares trançados da companhia telefônica, pois a maioria dos edifícios comerciais já está conectada dessa maneira, e normalmente há muitos pares sobressalentes disponíveis. Esse esquema é denominado **10Base-T.** Os hubs não armazenam no buffer o tráfego recebido. Descreveremos mais adiante neste capítulo uma versão aperfeiçoada dessa idéia (os switches) que guardam no buffer o tráfego recebido.

A Figura 4.14 mostra esses três esquemas de fiação. Para o 10Base5, um **transceptor** é preso firmemente ao cabo para que seu conector de pressão faça contato com o núcleo interno do cabo. O transceptor contém circuitos eletrônicos que tratam da detecção da portadora e da detecção de colisões. Quando é detectada uma colisão, o transceptor também injeta um sinal inválido especial no cabo, a fim de garantir que todos os outros transceptores também percebam que ocorreu uma colisão.

Com o 10Base5, um **cabo transceptor** ou **cabo de descida** conecta o transceptor a uma placa de interface no computador. O cabo transceptor pode ter até 50 m de comprimento e contém cinco pares trançados blindados individuais. Dois dos pares são destinados à entrada e à saída de dados, respectivamente. Dois outros são destinados a sinais de controle de entrada e saída. O quinto par, que nem sempre é utilizado, permite que o computador forneça energia aos circuitos do transceptor. Alguns transceptores permitem que até oito computadores vizinhos sejam conectados a ele, reduzindo assim o número de transceptores necessários.

[arte: imagem original da p. 273]

[Dísticos]

[1] Controlador

Cabo do transceptor

Derivação

[2] Núcleo

[3] Transceptor

(a)

[4] Transceptor + controlador

Conector

(b)

[5] Controlador

Par trançado

Hub

(c)

[F]Figura 4.14

[FL] Três tipos de cabeamento Ethernet. (a) 10Base5. (b) 10Base2. (c) 10Base-T

O cabo do transceptor termina na placa de interface dentro do computador. Essa placa contém um chip controlador que transmite quadros para o transceptor e recebe quadros dele. O controlador é responsável pela montagem dos dados em um formato de quadro apropriado, pelo cálculo de totais de verificação nos quadros enviados e nos quadros recebidos. Alguns chips controladores também gerenciam um grupo de buffers para quadros recebidos, uma fila de buffers para quadros a serem transmitidos, transferências diretas de memória com computadores hosts e outros aspectos do gerenciamento de rede.

Com o 10Base2, a conexão com o cabo consiste apenas em um conector BNC de junção em T. Os circuitos do transceptor estão localizados na placa controladora, e cada estação sempre tem seu próprio transceptor.

Com o 10Base-T, não existem cabos compartilhados, apenas o hub (uma caixa cheia de circuitos eletrônicos) ao qual cada estação é conectada por um cabo

dedicado (isto é, não compartilhado). A inclusão ou remoção de uma estação é mais simples nessa configuração, e os cabos partidos podem ser facilmente detectados. A desvantagem do 10Base-T é que o alcance máximo do cabo a partir do hub é de apenas 100 m, chegando talvez a 200 metros se forem usados pares trançados de qualidade muito alta da categoria 5. Mesmo assim, o 10Base-T vem se tornando cada vez mais popular, em virtude de sua facilidade de manutenção e do uso da fiação existente. Uma versão mais rápida do 10Base-T (100Base-T) será discutida mais adiante neste capítulo.

Uma quarta opção de cabeamento para Ethernet é o **10Base-F**, que utiliza fibra óptica. Essa alternativa é cara em função do custo dos conectores e terminadores, mas tem excelente imunidade a ruídos e representa o método preferido para edifícios ou hubs centrais muito distantes entre si. São permitidas distâncias de até 1 quilômetro. Ele também oferece boa segurança, pois é muito mais difícil montar derivações ("grampos") na fibra do que na fiação de cobre.

A Figura 4.15 mostra outras formas de instalar cabos em um edifício. Na Figura 4.15(a), um único cabo é arrastado de sala em sala, com cada estação se conectando a ele no ponto mais próximo. Na Figura 4.15(b), um backbone vertical vai do porão ao telhado, com cabos horizontais em cada andar ligados a ele por amplificadores especiais (repetidores). Em alguns edifícios, os cabos horizontais são finos e o backbone é grosso. A topologia mais comum é a de árvore, como mostra a Figura 4.15(c), pois uma rede com dois caminhos entre alguns pares de estações sofreria interferências entre os dois sinais.

[arte: imagem original da p. 274]

[Dísticos]

[1] A B

C

Derivação

D

(a)

[2] Backbone

(b)

[3] (c)

[4] A B C D

Repetidor

(d)

[F]Figura 4.15

[FL] Topologias de cabos. (a) Linear. (b) Em espinha. (c) Árvore. (d) Segmentada

Cada versão de Ethernet tem um comprimento máximo de cabo por segmento. Para permitir a conexão de redes maiores, vários cabos podem ser conectados por **repetidores**, como mostra a Figura 4.15(d). O repetidor é um dispositivo da camada física. Ele recebe, amplifica (regenera) e retransmite sinais em ambos os sentidos. No que se refere ao software, uma série de segmentos de cabos conectados por repetidores não é diferente de um único cabo (exceto pelo retardo introduzido pelos repetidores). Um sistema pode conter vários segmentos de cabos e repetidores, mas dois transceptores não podem estar a mais de 2,5 km de distância um do outro, e nenhum caminho entre dois transceptores quaisquer pode passar por mais de quatro repetidores.

[T3] 4.3.2 Codificação Manchester

Nenhuma das versões de Ethernet utiliza a codificação binária direta com 0 volts para representar um bit 0 e 5 volts para representar um bit 1, pois isso gera ambigüidades. Se uma estação enviar o string de bits 0001000, outras poderão interpretá-lo erradamente como 10000000 ou 01000000, pois não conseguem

identificar a diferença entre um transmissor inativo (0 volts) e um bit 0 (0 volts). Esse problema pode ser resolvido usando-se +1 volt para representar um bit 1 e -1 volt para representar um bit 0, mas ainda existe o problema de um receptor realizar a amostragem do sinal em uma freqüência um pouco diferente daquela que o transmissor usou para gerá-la. Diferentes velocidades do clock podem fazer o receptor e o transmissor tenderem à sincronização e não saberem onde estão os limites do bit, em especial após uma longa seqüência de valores 0 consecutivos ou de valores 1 consecutivos.

Tem de haver uma maneira de os receptores determinarem exatamente o início, o fim ou o meio de cada bit, sem fazer referência a um clock externo. Dois desses métodos são denominados **codificação Manchester** e **codificação Manchester diferencial.** Na codificação Manchester, cada período de bits é dividido em dois intervalos iguais. Um bit 1 binário é enviado quando a voltagem é definida como alta durante o primeiro intervalo, e como baixa no segundo intervalo. Um bit 0 binário é exatamente o oposto: primeiro baixo, e depois alto. Esse esquema garante que cada período de bit terá uma transição na parte intermediária, tornando fácil para o receptor sincronizar-se com o transmissor. Uma desvantagem da codificação Manchester é que ela exige duas vezes mais largura de banda que a codificação binária direta, pois os pulsos são a metade da largura. Por exemplo, para transmitir dados a 10 Mbps, o sinal tem de mudar 20 milhões de vezes por segundo. A codificação Manchester é mostrada na Figura 4.16(b).

[arte: imagem original da p. 275]

[Dísticos]

[1] Fluxo de bits 1 0 0 0 0 1 0 1 1 1 1

[2] (a) Codificação binária

[3] (b) Codificação Manchester

[4] (c) Codificação Manchester diferencial

[5] Aqui, a transição indica um valor 0

[6] Aqui, a falta de transição indica um valor 1

[F]Figura 4.16

[FL] (a) Codificação binária. (b) Codificação Manchester. (c) Codificação Manchester diferencial

A codificação Manchester diferencial, mostrada na Figura 4.16(c), é uma variação da codificação Manchester básica. Nela, um bit 1 é indicado pela ausência de uma transição no início do intervalo. Um bit 0 é indicado pela presença de uma transição no início do intervalo. Em ambos os casos, também existe uma transição na parte intermediária. O esquema diferencial exige equipamento mais complexo, mas oferece melhor imunidade a ruídos. Todos os sistemas Ethernet utilizam a codificação Manchester devido à sua simplicidade. O sinal alto tem +0,85 volts e o sinal baixo tem –0,85 volts, resultando em um valor de tensão CC igual a 0 volts. A Ethernet não utiliza a codificação Manchester diferencial, mas outras LANs (por exemplo, o token ring 802.5) o utilizam.

[T3] 4.3.3 O protocolo da subcamada MAC Ethernet

A estrutura original de quadros DIX (DEC, Intel, Xerox) é mostrada na Figura 4.17(a). Cada quadro começa com um *Preâmbulo* de 8 bytes, cada um contendo o padrão de bits 10101010. A codificação Manchester desse padrão produz uma onda quadrada de 10 MHz por 6,4 μs, a fim de permitir a sincronização entre o clock do receptor e o clock do transmissor. Eles devem permanecer sincronizados durante todo o restante do quadro, usando a codificação Manchester para controlar os limites de bits.

O quadro contém dois endereços, um para o destino e um para a origem. O

padrão permite endereços de 2 e de 6 bytes, mas os parâmetros definidos para o padrão de banda básica de 10 Mbps usam somente os endereços de 6 bytes. O bit de alta ordem do endereço de destino é 0 para endereços comuns e 1 para endereços de grupos. Os endereços de grupos permitem que diversas estações escutem um único endereço. Quando um quadro é enviado para um endereço de grupo, todas as estações do grupo o recebem. A transmissão para um grupo de estações é chamada de **multidifusão (multicast).** O endereço que consiste em todos os bits 1 é reservado para **difusão (broadcast).** Um quadro contendo todos os bits 1 no campo de destino é aceito por todas as estações da rede. A diferença entre multidifusão e difusão é importante o bastante para ser repetida. Um quadro de multidifusão é transmitido para um grupo selecionado de estações na Ethernet; um quadro de difusão é transmitido a todas as estações da Ethernet. A multidifusão é mais seletiva, mas envolve o gerenciamento de grupos. A difusão é menos seletiva, mas não requer qualquer gerenciamento de grupos.

[arte: imagem original da p. 276]

[Dísticos]

[1]Bytes 8 6 6 2 0 a 1500 0 a 46 4

(a) Preâmbulo Endereço de destino Endereço de origem Tipo

Dados Preenchimento Total de verificação

[2](b) Preâmbulo SOF Endereço de destino Endereço de origem

Comprimento Dados Preenchimento Total de verificação

[F]Figura 4.17

[FL] Formatos de quadros. (a) DIX Ethernet. (b) IEEE 802.3

Outra característica interessante do endereçamento é o uso do bit 46 (adjacente ao bit de mais alta ordem) para distinguir endereços locais de endereços globais. Os endereços locais são atribuídos pelo administrador da rede e não têm

significado fora da rede local. Os endereços globais, ao contrário, são atribuídos pelo IEEE para assegurar que duas estações de qualquer lugar do mundo nunca tenham o mesmo endereço global. Com os 48 – 2 = 46 bits disponíveis, existem cerca de 7×10^{13} endereços globais. A idéia é que qualquer estação possa endereçar exclusivamente qualquer outra estação, fornecendo apenas o número de 48 bits correto. Cabe à camada da rede descobrir como localizar o destino.

Em seguida, vem o campo *Tipo*, que informa ao receptor o que fazer com o quadro. Vários protocolos da camada de rede podem estar em uso ao mesmo tempo na mesma máquina; assim, ao chegar um quadro Ethernet, o núcleo tem de saber a qual deles deve entregar o quadro. O campo *Tipo* especifica que processo deve receber o quadro.

Depois, vêm os dados, com até 1500 bytes. Esse limite foi escolhido de forma um tanto arbitrária na época em que o padrão DIX foi esculpido em pedra, principalmente com base no fato de que um transceptor precisa ter RAM suficiente para guardar um quadro inteiro e, em 1978, a RAM tinha um custo muito alto. Um limite superior maior significaria mais RAM e, conseqüentemente, um transceptor mais caro.

Além de haver um comprimento máximo de quadro, também existe um comprimento mínimo de quadro. Embora um campo de dados de 0 bytes às vezes seja útil, ele causa um problema. Quando detecta uma colisão, um transceptor trunca o quadro atual, o que significa que bits perdidos e fragmentos de quadros aparecem a todo instante no cabo. Para tornar mais fácil a distinção entre quadros válidos e lixo, o padrão Ethernet exige que os quadros válidos tenham pelo menos 64 bytes de extensão, do endereço de destino até o campo de total de verificação, incluindo ambos. Se a parte de dados de um quadro for menor que 46 bytes, o campo *Preenchimento* será usado para preencher o quadro até o tamanho mínimo.

Outra (e mais importante) razão para a existência de um quadro de comprimento mínimo é impedir que uma estação conclua a transmissão de um quadro curto antes do primeiro bit ter atingido a outra extremidade do cabo, onde ele poderá colidir com outro quadro. Esse problema é ilustrado na Figura 4.18. No tempo 0, a estação *A* — localizada em uma extremidade da rede — envia um quadro. Vamos chamar o tempo de propagação que esse quadro leva para atingir a outra extremidade [ver símbolo]. Momentos antes do quadro chegar à outra extremidade (ou seja, no tempo [ver símbolo]), a estação mais distante, *B*, inicia a transmissão. Quando detecta que está recebendo mais potência do que está transmitindo, *B* sabe que ocorreu uma colisão, interrompe a transmissão e gera uma rajada de ruído de 48 bits para avisar a todas as outras estações. Em outras palavras, ela bloqueia o éter para ter certeza de que o transmissor não irá ignorar a colisão. Aproximadamente no instante [ver símbolo], o transmissor detecta a rajada de ruído e também interrompe sua transmissão. Em seguida, ele aguarda um intervalo de tempo aleatório antes de tentar novamente.

[arte: imagem original da p. 277]

[Dísticos]

[1] A O pacote começa no tempo 0 B

(a)

[2] A Pacote quase em B no instante [ver símbolo] B

(b)

[3] A B

(c) Colisão no tempo [ver símbolo]

[4] A Rajada de ruído retorna para A no tempo [ver símbolo] B

(d)

[F]Figura 4.18

[FL] A detecção de colisão pode demorar até o tempo [ver símbolo]

Se uma estação tentar transmitir um quadro muito curto, é concebível que ocorra uma colisão. No entanto, mesmo assim, a transmissão será concluída antes que a rajada de ruído retorne no instante [ver símbolo]. Então, o transmissor concluirá incorretamente que o quadro foi enviado com êxito. Para evitar que essa situação ocorra, a transmissão de todos os quadros deve demorar mais de [ver símbolo] para ser concluída, de forma que a transmissão ainda esteja acontecendo quando a rajada de ruído voltar ao transmissor. Para uma LAN de 10 Mbps com um comprimento máximo de 2500 metros e quatro repetidores (de acordo com a especificação 802.3), o tempo de ida e volta (incluindo o tempo de propagação pelos quatro repetidores) foi calculado em quase 50 µs no pior caso, incluindo o tempo para a passagem pelos quatro repetidores que, sem dúvida, não é igual a zero. Portanto, o quadro mínimo deve demorar pelo menos esse tempo para ser transmitido. A 10 Mbps, um bit demora 100 ns, e assim 500 bits é o menor tamanho de quadro que oferece a garantia de funcionar. Para acrescentar uma certa margem de segurança, esse número foi arredondado para 512 bits ou 64 bytes. Quadros com menos de 64 bytes são preenchidos até completar 64 bytes com o campo *Preenchimento*.

À medida que a velocidade da rede cresce, o comprimento mínimo de quadro deve aumentar ou o comprimento máximo de cabo deve diminuir proporcionalmente. Para uma LAN de 2500 metros operando a 1 Gbps, o tamanho mínimo de quadro teria de ser de 6400 bytes. Como alternativa, o tamanho mínimo de quadro poderia ser de 640 bytes, e a distância máxima entre duas estações quaisquer poderia ser de 250 m. Essas restrições estão se tornando cada vez mais penosas, à medida que migramos em direção às redes de vários gigabits.

O último campo Ethernet é o *Total de verificação*. Ele é efetivamente um código

de hash de 32 bits dos dados. Se alguns bits de dados forem recebidos com erros (devido ao ruído no cabo), o total de verificação quase certamente estará errado, e o erro será detectado. O algoritmo do total de verificação é um CRC (Cyclic Redundancy Check) do tipo descrito no Capítulo 3. Ele simplesmente realiza a detecção de erros, não a correção de erros antecipada.

Quando o IEEE padronizou a Ethernet, o comitê fez duas alterações no formato DIX, como mostra a Figura 4.17(b). A primeira foi reduzir o preâmbulo para 7 bytes e usar o último byte como um delimitador de *Início de quadro*, por compatibilidade com os padrões 802.4 e 802.5. A segunda alteração foi transformar o campo *Tipo* em um campo *Comprimento*. É claro que nesse caso não havia nenhum modo para o receptor descobrir o que fazer com um quadro recebido, mas esse problema foi tratado com a inclusão de um pequeno cabeçalho na própria porção de dados, a fim de fornecer essa informação. Descreveremos o formato da porção de dados quando chegarmos ao controle de enlace lógico, mais adiante neste capítulo.

Infelizmente, quando o 802.3 foi publicado, já havia tanto hardware e software para Ethernet DIX em uso que poucos fabricantes e usuários ficaram entusiasmados com a possibilidade de converter o campo *Tipo* em um campo *Comprimento*. Em 1997, o IEEE desistiu e afirmou que ambos os formatos eram bons para ele. Felizmente, todos os campos *Tipo* em uso antes de 1997 tinham mais de 1500 bytes. Como conseqüência, qualquer número contido nesse campo que seja menor que ou igual a 1500 pode ser interpretado como *Comprimento*, e qualquer número maior que 1500 pode ser interpretado como *Tipo*. Agora, o IEEE pode afirmar que todo mundo está usando seu padrão e que qualquer pessoa pode continuar a fazer o que já estava fazendo sem se sentir culpado.

[T3] 4.3.4 O algoritmo de recuo binário exponencial

Vejamos agora como é feita a randomização quando ocorre uma colisão. O modelo é o da Figura 4.5. Depois de uma colisão, o tempo é dividido em slots discretos, cujo comprimento é igual ao pior tempo de propagação de viagem de ida e volta no éter ([ver símbolo]). Para acomodar o caminho mais longo permitido pelo padrão Ethernet, o tempo de duração do slot foi definido como 512 períodos de duração de um bit, ou 51,2 μs, conforme mencionamos antes. Depois da primeira colisão, cada estação espera 0 ou 1 tempos de slot antes de tentar novamente. Se duas estações colidirem e selecionarem o mesmo número aleatório, elas colidirão novamente. Depois da segunda colisão, cada uma seleciona ao acaso 0, 1, 2 ou 3 e aguarda durante esse número de tempos de slot. Se ocorrer uma terceira colisão (cuja probalidade é de 0,25), na próxima vez o número de slots que a estação deverá esperar será escolhido ao acaso no intervalo de 0 a $2^3 - 1$.

Em geral, depois de i colisões, é escolhido um número aleatório entre 0 e $2^i - 1$, e esse número de slot será ignorado. Entretanto, após terem sido alcançadas dez colisões, o intervalo de randomização será congelado em um máximo de 1023 slots. Depois de 16 colisões, o controlador desiste e informa o erro ao computador. Qualquer recuperação adicional caberá às camadas superiores.

Esse algoritmo, chamado **recuo binário exponencial**, foi escolhido para se adaptar dinamicamente ao número de estações que estão tentando transmitir. Se o intervalo de escolha do número aleatório para todas as colisões fosse 1023, a chance de duas estações colidirem uma segunda vez seria desprezível, mas o tempo de espera médio depois de uma colisão seria de centenas de períodos de slot, introduzindo um retardo significativo. Por outro lado, se cada estação sempre esperasse durante 0 ou 1 slot, e se 100 estações tentassem transmitir ao mesmo tempo, elas colidiriam repetidas vezes até que 99 delas escolhessem 1 e a estação restante escolhesse 0. Isso poderia levar anos. Aumentando-se expo-

nencialmente o intervalo de randomização à medida que ocorre um número cada vez maior de colisões consecutivas, o algoritmo assegura um baixo retardo quando apenas algumas estações colidem, mas também garante que a colisão será resolvida em um intervalo de tempo razoável quando muitas estações colidirem. A restrição do recuo a 1023 impede que o limite cresça demais.

Como descrevemos até agora, o CSMA/CD não fornece nenhuma confirmação. Como a simples ausência de colisões não garante que os bits não foram adulterados por picos de ruído no cabo, para obter uma comunicação confiável, o destino deve conferir o total de verificação e, se ele estiver correto, deve transmitir um quadro de confirmação para a origem. Normalmente, essa confirmação seria apenas outro quadro no que se refere ao protocolo, e teria de disputar o tempo do canal, assim como qualquer outro quadro de dados. Contudo, uma simples modificação no algoritmo de disputa permitiria uma confirmação mais rápida da recepção do quadro (Tokoro e Tamaru, 1977). Seria necessário apenas reservar o primeiro slot de disputa após cada transmissão bem-sucedida para a estação de destino. Infelizmente, o padrão não oferece essa possibilidade.

[T3] 4.3.5 Desempenho da Ethernet

Agora, vamos examinar rapidamente o desempenho da Ethernet sob condições de carga alta e constante, ou seja, k estações sempre prontas a transmitir. Uma análise completa do algoritmo de recuo binário exponencial é muito complicada. Em vez disso, seguiremos Metcalfe e Boggs (1976) e iremos supor uma probabilidade de retransmissão constante em cada slot. Se cada estação transmitir durante um slot de disputa com probabilidade p, a probabilidade A de que alguma estação tome posse do canal existente nesse slot será:

[Inserir equação do O.A. p. 279] (4-5)

A é maximizado quando $p = 1/k$, com $A \to 1/e$, à medida que $k \to \infty$. A probabilidade de que o intervalo de disputa tenha exatamente j slots é $A(1 - A)^{j-1}$, de forma que o número médio de slots por disputa é dado por:

[Inserir equação do O.A. p. 280a]

Como cada slot tem a duração [ver símbolo], o intervalo médio de disputa, w, é [ver símbolo]. Supondo-se um valor ótimo para p, o número médio de slots de disputa nunca é maior que e; portanto, w é no máximo [ver símbolo].

Se o quadro médio leva P segundos para ser transmitido, quando muitas estações têm quadros a enviar, temos:

[Inserir equação do O.A. p. 280b] (4-6)

(a) Eficiência do canal

Aqui, vemos que a distância máxima do cabo entre duas estações entra nos números do desempenho, dando origem a outras topologias diferentes das tipologias mostradas na Figura 4.15(a). Quanto maior for o cabo, maior será o intervalo de disputa. Essa observação explica por que o padrão Ethernet especifica um comprimento máximo de cabo.

É instrutivo formular a Equação (4-6) em termos do comprimento do quadro, F, da largura de banda da rede, B, do comprimento do cabo, L e da velocidade de propagação do sinal, c, para o caso ótimo de e slots de disputa por quadro. Com $P = F/B$, a Equação (4-6) passa a ser:

[Inserir equação do O.A. p. 280c] (4-7)

(a) Eficiência do canal

Quando o segundo termo no denominador for grande, a eficiência da rede será baixa. Mais especificamente, aumentar a largura de banda da rede ou a distância (o produto BL) reduz a eficiência para um determinado tamanho de quadro. Infelizmente, a maior parte das pesquisas em hardware de rede visa exatamente ao aumento desse produto. As pessoas querem alta largura de banda em longas

distâncias (MANs de fibra óptica, por exemplo), o que sugere que o padrão Ethernet implementado dessa maneira talvez não seja o melhor sistema para essas aplicações. Veremos outras formas de implementar a Ethernet quando estudarmos a Ethernet comutada, mais adiante neste capítulo.

Na Figura 4.19, a eficiência do canal é representada contra o número de estações prontas para [ver símbolo] e uma taxa de dados de 10 Mbps, usando-se a Equação (4-7). Com um tempo de slot de 64 bytes, não surpreende que quadros de 64 bytes não sejam eficientes. Por outro lado, com quadros de 1024 bytes e um valor assintótico de *e* slots de 64 bytes por intervalo de disputa, o período de disputa é de 174 bytes e a eficiência é 0,85.

Para determinar o número médio de estações prontas para transmitir sob condições de alta carga, podemos usar a seguinte observação (grosso modo). Cada quadro ocupa o canal por um período de disputa e um tempo de transmissão de quadro, resultando em um total de $P + w$ segundos. O número de quadros por segundo é portanto $1/(P + w)$. Se cada estação gerar quadros a uma taxa média de λ quadros/s, quando o sistema estiver no estado k, a taxa de entrada total de todas as estações desbloqueadas combinadas será de $k\lambda$ quadros/s. Tendo em vista que, no estado de equilíbrio, as taxas de entrada e saída devem ser idênticas, podemos igualar essas duas expressões e resolver a equação para k. (Observe que w é uma função de k.) Uma análise mais sofisticada é apresentada em (Bertsekas e Gallager, 1992).

[arte: imagem original da p. 281]

[Dísticos]

[1] Eficiência do canal

1,0

0,9

0,8

0,7

0,6

0,5

0,4

0,3

0,2

0,1

[2]Quadros de 1024 bytes

Quadros de 512 bytes

Quadros de 256 bytes

Quadros de 128 bytes

Quadros de 64 bytes

[3]0 1 2 4 8 16 32 64 128 256

[4] Número de estações tentando transmitir

[F]Figura 4.19

[FL] Eficiência da Ethernet a 10 Mbps com tempos de slot de 512 bits

Talvez falha a pena mencionar que houve um grande número de análises teóricas sobre o desempenho da Ethernet (e de outras redes). Praticamente todos esses trabalhos presumiram que o tráfego obedece a uma série de Poisson. Como os pesquisadores começaram a analisar dados reais, parece que agora o tráfego de rede raras vezes é de Poisson, mas é semelhante (Paxson e Floyd, 1994; e Willinger *et al.*, 1995). Isso significa que calcular uma média durante intervalos de tempo longos não suaviza o tráfego. O número médio de quadros em cada minuto de uma hora possui a mesma variação que o número médio de quadros em cada segundo de um minuto. A conseqüência dessa descoberta é que a maioria dos modelos de tráfego de rede não se aplica ao mundo real, e deve ser

tomada com grande restrição.

[T3] 4.3.6 Ethernet comutada

À medida que mais e mais estações são acrescentadas a uma rede Ethernet, o tráfego aumenta. Eventualmente, a LAN ficará saturada. Uma saída é aumentar a velocidade, digamos, de 10 Mbps para 100 Mbps. Porém, com o crescimento da multimídia, até mesmo uma rede Ethernet de 100 Mbps ou 1 Gbps pode se tornar saturada.

Felizmente, existe uma outra solução menos drástica para lidar com o aumento da carga: a Ethernet comutada, ilustrada na Figura 4.20. O núcleo desse sistema é um **switch**, que contém um @@@backplane de alta velocidade e espaço para 4 a 32 placas de linha plug-in, cada uma contendo de 1 a 8 conectores. Com freqüência, cada conector tem uma conexão de par trançado 10Base-T com um único computador host.

[arte: imagem original da p.]

[Dísticos]

[1] Conector

[2] Switch

[3] Para hosts

Ethernet Hub Para hosts

Para hosts

[4] Conexão 10Base-T

[5] Para os computadores hosts

[F]Figura 4.20

[FL] Um exemplo simples de Ethernet comutada

Quando deseja transmitir um quadro Ethernet, a estação envia um quadro padrão

para o switch. A placa plug-in que obtém o quadro verifica se ele se destina a uma das outras estações conectadas à mesma placa. Se esse for o caso, o quadro será copiado. Do contrário, o quadro será enviado pelo backplane de alta velocidade para a placa da estação de destino. Em geral, o backplane é executado a mais de 1 Gbps quando utiliza um protocolo patenteado.

O que acontecerá se duas máquinas conectadas à mesma placa plug-in transmitirem quadros ao mesmo tempo? Isso depende da forma como a placa foi elaborada. Uma possibilidade é que todas as portas da placa estejam fisicamente conectadas por fios, de modo a formar uma LAN local na placa. As colisões dessa LAN na placa serão detectadas e tratadas da mesma forma que qualquer outra colisão em uma rede CSMA/CD — com as retransmissões usando um algoritmo de recuo binário exponencial. Com esse tipo de placa plug-in, só é possível uma transmissão por placa em um determinado momento, mas todas as placas podem transmitir em paralelo. Com esse projeto, cada placa forma seu próprio **domínio de colisão**, independente das outras. Com apenas uma estação por domínio de colisão, as colisões são impossíveis, e o desempenho é otimizado.

Com o outro tipo de placa plug-in, cada porta de entrada é mantida em um buffer. Sendo assim, os quadros recebidos são armazenados na RAM on-board da placa, à medida que chegam. Esse projeto permite que todas as portas de entrada recebam (e transmitam) quadros ao mesmo tempo, em operação paralela, full-duplex. Quando um quadro é completamente recebido, a placa pode verificar se o quadro é destinado a outra porta da mesma placa, ou a uma porta distante. No primeiro caso, o quadro é transmitido diretamente para o destino. No segundo caso, o quadro deve ser transmitido pelo backplane até a placa correta. Com esse projeto, cada porta é um domínio de colisão separado, impedindo a ocorrência de colisões. Com freqüência, o throughput total do sistema pode ser aumentado em uma ordem de magnitude em relação ao 10Base5, que tem um único domínio de

colisão para o sistema inteiro.

Tendo em vista que o switch espera apenas quadros Ethernet padrão em cada porta de entrada, é possível usar algumas dessas portas como concentradores. Na Figura 4.20, a porta localizada no canto superior direito não está conectada a uma estação isolada, mas a um hub de 12 portas. À medida que chegam ao hub, os quadros disputam a rede Ethernet da forma usual, inclusive com colisões e recuo binário. Os quadros bem-sucedidos são enviados ao switch e são tratados como quaisquer outros quadros recebidos: eles são comutados para a linha de saída correta através do backplane de alta velocidade. Os hubs são mais econômicos que os switches mas, devido à queda nos preços dos switches, eles estão se tornando obsoletos rapidamente. Apesar disso, ainda existem hubs de tecnologia antiga.

[T3] 4.3.7 Fast Ethernet

A princípio, 10 Mbps parecia ser o paraíso, da mesma forma que os modems de 1200 bps pareciam ser o paraíso para os primeiros usuários de modems acústicos de 300 bps. Porém, a novidade se dissipou com rapidez. Como uma espécie de corolário da Lei de Parkinson ("O trabalho se expande até preencher o tempo disponível para sua conclusão"), parecia que os dados se expandiam para preencher toda a largura de banda disponível para sua transmissão. Para aumentar a velocidade, vários grupos industriais propuseram duas novas LANs ópticas baseadas em anel. Uma foi chamada **FDDI (Fiber Distributed Data Interface — interface de dados distribuída por fibra)** e a outra foi chamada **Fibre Channel**[*]. Para encurtar a história, embora ambas fossem usadas como redes de backbone, nenhuma delas teve amplo sucesso. Em ambos os casos, o gerenciamento da estação era muito complicado, o que levou a chips complexos e preços elevados. A lição a ser tirada de tudo isso é que devemos manter a

máxima simplicidade possível.

[Nota de rodapé]

* Essa LAN foi chamada "fibre channel", e não "fiber channel" porque o editor do documento era britânico.

De qualquer modo, o fato das LANs ópticas não terem se tornado populares deixou um imenso espaço para uma grande variedade de redes Ethernet com velocidades acima de 10 Mbps. Muitas instalações precisavam de maior largura de banda e tinham diversas LANs de 10 Mbps conectadas por um labirinto de repetidores, pontes, roteadores e gateways, embora às vezes parecesse para os administradores de redes que elas estavam conectadas por goma de mascar e tela de arame.

Foi nesse ambiente que o IEEE reuniu o comitê do 802.3 em 1992, com instruções para produzir uma LAN mais rápida. Uma das propostas era manter o 802.3 exatamente como estava, e apenas torná-lo mais rápido. Outra proposta era refazê-lo completamente, para integrar um grande número de novos recursos, como tráfego em tempo real e voz digitalizada, mas manter o antigo nome (por motivos de marketing). Após alguma discussão, o comitê decidiu manter o 802.3 como ele era, simplesmente tornando-o mais rápido. As pessoas que apoiavam a proposta perdedora fizeram o que qualquer pessoa do setor de informática faria nessas circunstâncias — formaram seu próprio comitê e padronizaram sua LAN mesmo assim (eventualmente, como o padrão 802.12). Esse padrão fracassou por completo.

As três principais razões pelas quais o comitê do 802.3 decidiu continuar com uma rede Ethernet aperfeiçoada foram:

1. A necessidade de manter a compatibilidade retroativa com as LANs Ethernet existentes.

2. O medo de que um novo protocolo criasse problemas imprevistos.

3. O desejo de terminar o trabalho antes que a tecnologia mudasse.

O trabalho foi feito rapidamente (pelas normas dos comitês de padronização) e o resultado, o **802.3u**, foi oficialmente aprovado pelo IEEE em junho de 1995. Tecnicamente, o 802.3u não é um padrão novo, mas um adendo ao padrão 802.3 existente (para enfatizar sua compatibilidade retroativa). Como todos o chamam **Fast Ethernet**, em vez de 802.3u, também faremos o mesmo.

A idéia básica por trás do Fast Ethernet era simples: manter os antigos formatos de quadros, interfaces e regras de procedimentos, e apenas reduzir o tempo de bit de 100 ns para 10 ns. Tecnicamente, teria sido possível copiar o 10Base-5 ou o 10Base-2 e continuar a detectar colisões a tempo, pela simples redução do comprimento máximo do cabo a um décimo do comprimento original. Entretanto, as vantagens do cabeamento 10Base-T eram tão grandes que o Fast Ethernet se baseou inteiramente nesse projeto. Por isso, todos os sistemas Fast Ethernet usam hubs e switches; cabos multiponto com conectores de pressão ou conectores BNC não são permitidos.

Entretanto, algumas decisões ainda precisavam ser tomadas, sendo a mais importante delas os tipos de fios que seriam aceitos. Um dos concorrentes era o par trançado da categoria 3. O argumento a favor dele era que todo escritório do mundo ocidental tinha pelo menos quatro pares trançados da categoria 3 (ou melhor) instalados entre ele e um armário de fiação telefônica a uma distância máxima de 100 metros. Às vezes, há dois cabos desse tipo. Desse modo, o uso do par trançado da categoria 3 tornaria possível conectar computadores de desktop com o emprego de Fast Ethernet, sem a necessidade de refazer a fiação do edifício, uma enorme vantagem para muitas empresas.

A principal desvantagem do par trançado da categoria 3 é sua incapacidade para transportar sinais de 200 megabauds (100 Mbps com codificação Manchester) por 100 metros, a distância máxima entre o computador e o hub especificada para

10Base-T (ver Figura 4.13). Por outro lado, a fiação de par trançado da categoria 5 é capaz de tratar 100 metros com facilidade, e a fibra pode ir muito mais longe que isso. Decidiu-se permitir as três possibilidades, como mostra a Figura 4.21, mas incentivar a solução da categoria 3, para que fosse possível obter a capacidade de transporte adicional necessária.

[arte: ver original p. 284]

[Tabela]

Nome	Cabo	Tam. máx. de segmento	Vantagens
100Base-T4	Par trançado	100 m	Utiliza UTP da categoria 3
100Base-TX	Par trançado	100 m	Full-duplex a 100 Mbps (UTP da categoria 5)
100Base-FX	Fibra óptica	2000 m	Full-duplex a 100 Mbps; grandes distâncias

[F]Figura 4.21

[FL] O cabeamento Fast Ethernet

O esquema UTP (unshielded twisted pair — par trançado sem blindagem) da categoria 3, chamado **100Base-T4**, emprega uma velocidade de sinalização de 25 MHz, somente 25% mais rápida do que os 20 MHz da Ethernet padrão (lembre-se de que a codificação Manchester, mostrada na Figura 4.16, requer dois períodos de clock para cada um dos 10 milhões de bits, a cada segundo). Porém, para atingir a largura de banda necessária, o 100Base-T4 exige quatro pares trançados. Como a fiação telefônica padrão teve quatro pares trançados por cabo durante décadas, a maioria dos escritórios é capaz de lidar com esse requisito. É claro que isso significa abrir mão do seu telefone comercial, mas sem dúvida trata-se de um pequeno preço a ser pago por um serviço de correio eletrônico mais rápido.

Dos quatro pares trançados, um é sempre destinado ao hub, um sempre vem do hub, e os outros dois são comutáveis no sentido em que estiver sendo realizada a transmissão. Para obter a largura de banda necessária, a codificação Manchester não é utilizada; no entanto, com clocks modernos e distâncias curtas como essas, ela já não é mais necessária. Além disso, são enviados sinais ternários; assim, durante um único período de clock, o fio pode conter um valor 0, um valor 1 ou um valor 2. Com três pares trançados orientados no sentido direto e a sinalização ternária, pode-se transmitir qualquer um dos 27 símbolos possíveis, o que torna viável a transmissão de 4 bits com alguma redundância. A transmissão de 4 bits em cada um dos 25 milhões de ciclos de clock por segundo fornece os 100 Mbps necessários. Além disso, há sempre um canal reverso de 33,3 Mbps que utiliza o par trançado restante. Esse esquema, conhecido como **8B/6T** (8 bits mapeados em 6 trits), provavelmente não será premiado por elegância, mas funciona com o esquema de fiação existente.

Para a fiação da categoria 5, o projeto **100Base-TX** é mais simples, porque os fios são capazes de manipular velocidades do clock de até 125 MHz. São usados somente dois pares trançados por estação, um que vai para o hub e outro que sai do hub. Em vez de usar apenas a codificação binária direta, é usado um esquema chamado **4B/5B**. Esse esquema se baseia no FDDI e é compatível com ele. Cada grupo de cinco períodos de clock, contendo um entre dois valores de sinais, produz 32 combinações. Dezesseis dessas combinações são usadas para transmitir os grupos de 4 bits 0000, 0001, 0010, ..., 1111. Algumas das dezesseis combinações restantes são utilizadas para fins de controle, como a demarcação dos limites dos quadros. As combinações empregadas foram escolhidas com todo o cuidado, a fim de fornecerem transições suficientes para manter a sincronização do clock. O 100Base-TX é um sistema full-duplex; as estações podem transmitir a 100 Mbps e receber a 100 Mbps, ao mesmo tempo.

Com freqüência, o 100Base-TX e o 100Base-T4 são referidos em conjunto como **100Base-T**.

A última opção, o **100Base-FX**, utiliza dois filamentos de fibra multimodo, um para cada sentido; por isso, ele também é full-duplex, com 100 Mbps em cada sentido. Além disso, a distância entre uma estação e o hub pode ser de até 2 km.

Em resposta à demanda popular, em 1997, o comitê 802 acrescentou um novo tipo de cabeamento, o 100Base-T2, que permite à Fast Ethernet funcionar em dois pares de fios existentes da categoria 3. No entanto, é necessário um sofisticado processador de sinais digitais para lidar com o esquema de codificação exigido, o que torna essa opção bastante dispendiosa. Até agora, ela raramente é usada devido à sua complexidade, a seu custo e ao fato de que muitos edifícios de escritórios já tiveram sua fiação trocada por UTP da categoria 5.

São possíveis dois tipos de dispositivos de interconexão com o 100Base-T: hubs e switches, como mostra a Figura 4.20. Em um hub, todas as linhas de entrada (ou pelo menos todas as linhas que chegam a uma placa plug-in) estão logicamente conectados, formando um único domínio de colisão. Aplicam-se todas as regras padrão, inclusive o algoritmo de recuo binário exponencial, e assim o sistema funciona da mesma forma que o antigo padrão Ethernet. Em particular, apenas uma estação pode transmitir de cada vez. Em outras palavras, os hubs exigem comunicação half-duplex.

Em um switch, cada quadro de entrada é armazenado no buffer em uma placa de linha plug-in e repassado por um backplane de alta velocidade da placa de origem à placa de destino, se necessário. O backplane não foi padronizado, nem precisa ser, pois ele fica inteiramente oculto no interior do switch. Se a experiência anterior servir de guia, os fornecedores de switches irão competir com vigor para produzir backplanes cada vez mais rápidos, a fim de melhorar o

throughput do sistema. Como os cabos 100Base-X são longos demais para o algoritmo normal de colisões Ethernet, eles devem ser conectados a switches, de modo que cada um seja um domínio de colisão em si mesmo. Os hubs não são permitidos no padrão 100Base-FX.

Para finalizar, praticamente todos os switches podem manipular uma mistura de estações de 10 Mbps e 100 Mbps, para facilitar a atualização. À medida que um site adquirir mais e mais estações de trabalho de 100 Mbps, ele só precisará comprar o número necessário de novas placas de linha e inseri-las no switch. Na verdade, o próprio padrão oferece um meio para duas estações negociarem de modo automático a velocidade ótima (10 ou 100 Mbps) e o tipo de comunicação (half-duplex, ou full-duplex). A maioria dos produtos de Fast Ethernet utiliza esse recurso para realizar sua própria configuração automática.

[T3] 4.3.8 Ethernet de gigabit

A tinta mal havia secado no padrão Fast Ethernet quando o comitê 802 começou a trabalhar em uma Ethernet ainda mais rápida (1995). Ele foi denominado **Ethernet de gigabit** e foi ratificado pelo IEEE em 1998, com o nome 802.3z. Esse identificador sugere que a Ethernet de gigabit será o final da linha, a menos que alguém invente uma nova letra depois de z. Descreveremos a seguir algumas das principais características da Ethernet de gigabit. Você poderá encontrar mais informações em (Seifert, 1998).

Os objetivos do comitê do 802.3z eram essencialmente os mesmos do comitê 802.3u: tornar a Ethernet 10 vezes mais rápida, mantendo a compatibilidade retroativa com todos os padrões Ethernet existentes. Em particular, a Ethernet de gigabit tinha de oferecer o serviço de datagrama não confirmado com unidifusão e multidifusão, empregar o mesmo esquema de endereçamento de 48 bits já em uso e manter o mesmo formato de quadro, inclusive os tamanhos mínimo e

máximo de quadro. O padrão final atendeu a todos esses objetivos.

Todas as configurações de Ethernet de gigabit são ponto a ponto, e não multiponto como no padrão original de 10 Mbps, agora honrado como o título de **Ethernet clássica.** Na configuração mais simples de Ethernet de gigabit, ilustrada na Figura 4.22(a), dois computadores estão diretamente conectados um ao outro. Porém, o caso mais comum consiste em um switch ou um hub conectado a vários computadores e possivelmente a switches ou hubs adicionais, como mostra a Figura 4.22(b). Em ambas as configurações, cada cabo Ethernet individual tem exatamente dois dispositivos conectados a ele, nem mais nem menos.

[arte: imagem original da p. 287]

[Dísticos]

[1] Ethernet

Computador

(a)

[2] Switch ou hub

Ethernet

(b)

[F]Figura 4.22

[FL] (a) Uma Ethernet de duas estações. (b) Uma Ethernet de várias estações

A Ethernet de gigabit admite dois modos de operação diferentes: o modo full-duplex e o modo half-duplex. O modo "normal" é o modo full-duplex, que permite tráfego em ambos os sentidos ao mesmo tempo. Esse modo é usado quando existe um switch central conectado a computadores (ou outros switches) na periferia. Nessa configuração, todas as linhas são armazenadas no buffer, de forma que cada computador e cada switch é livre para enviar quadros sempre que quiser. O transmissor não tem de detectar o canal para saber se ele está sendo

usado por mais alguém, porque a disputa é impossível. Na linha entre um computador e um switch, o computador é o único transmissor possível para o switch naquela linha, e a transmissão tem sucesso ainda que o switch esteja transmitindo no momento um quadro para o computador (porque a linha é full-duplex). Tendo em vista que não é possível nenhuma disputa, o protocolo CSMA/CD não é usado, e assim o comprimento máximo do cabo é determinado pela intensidade do sinal, e não pelo tempo que uma rajada de ruído leva para se propagar de volta até o transmissor no pior caso. Os switches são livres para se misturar e equipar suas velocidades. A configuração automática é admitida, como na Fast Ethernet.

O outro modo de operação, o half-duplex, é usado quando os computadores estão conectados a um hub, e não a um switch. Um hub não armazena os quadros recebidos do buffer. Em vez disso, ele estabelece conexões elétricas internas para todas as linhas, simulando o cabo multiponto usado na Ethernet clássica. Nesse modo, são possíveis colisões e, portanto, é necessário o protocolo CSMA/CD padrão. Tendo em vista que um quadro mínimo (isto é, de 64 bytes) agora pode ser transmitido 100 vezes mais rápido que na Ethernet clássica, a distância máxima é 100 vezes menor (ou seja, 25 metros), a fim de manter a propriedade essencial de que o transmissor ainda irá transmitir quando a rajada de ruído voltar a ele, mesmo no pior caso. Com um cabo de 2500 metros, o transmissor de um quadro de 64 bytes a 1 Gbps terminaria a transmissão bem antes do quadro sequer ter chegado a percorrer um décimo da distância até a outra extremidade, quanto mais ir até a extremidade e voltar.

O comitê 802.3z considerou um raio de 25 metros inaceitável e acrescentou duas características ao padrão para aumentar o raio. A primeira característica, chamada **extensão de portadora**, essencialmente informa ao hardware para adicionar seu próprio preenchimento ao quadro normal, a fim de estender o

quadro a 512 bytes. Tendo em vista que esse preenchimento é adicionado pelo hardware transmissor e removido pelo hardware receptor, o software não tem conhecimento desse fato, o que significa que não é necessária nenhuma mudança no software existente. É claro que o uso de 512 bytes de largura de banda para transmitir 46 bytes de dados do usuário (a carga útil de um quadro de 64 bytes) tem uma eficiência de linha igual a 9%.

A segunda característica, chamada **rajada de quadros**, permite a um transmissor enviar uma seqüência concatenada de vários quadros em uma única transmissão. Se a rajada total tiver menos de 512 bytes, o hardware a preencherá novamente. Se houver quadros suficientes esperando pela transmissão, esse esquema será altamente eficiente e preferível à extensão de portadora. Essas novas características estendem o raio da rede a 200 metros, o que deve ser suficiente para a maioria dos escritórios.

Com toda franqueza, é difícil imaginar uma organização se envolvendo com as dificuldades de compra e instalação de placas Ethernet de gigabit para obter alto desempenho, e depois conectar os computadores a um hub para simular a Ethernet clássica, com todas as suas colisões. Embora os hubs sejam um pouco mais econômicos que os switches, as placas de interface da Ethernet de gigabit ainda são relativamente dispendiosas. Assim, economizar comprando um hub de baio custo e reduzir o desempenho do novo sistema é tolice. Ainda assim, a compatibilidade retroativa é sagrada na indústria de informática, e então o comitê do 802.3z foi obrigado a aceitá-lo.

A Ethernet de gigabit admite cabeamento de cobre e de fibra, como mostra a Figura 4.23. A sinalização à velocidade de aproximadamente 1 Gbps sobre fibra significa que a fonte de luz tem de ser ligada e desligada dentro do intervalo de 1 ns. Os leds simplesmente não podem operar com tanta rapidez, e assim são necessários lasers. São permitidos dois comprimentos de onda: 0,85 micra (curto)

e 1,3 micra (longo). Os lasers a 0,85 micra têm custo mais baixo, mas não funcionam em fibra de modo único.

[arte: ver original p. 288]

[Tabela]

Nome	**Cabo**	**Tam. máx. de segmento**	**Vantagens**
1000Base-SX	Fibra óptica	550 m	Fibra de multimodo (50, 62,5 micra)
1000Base-LX	Fibra óptica	5000 m	Modo único (10 µ) ou multimodo (50, 62,5 µ)
1000Base-CX	2 pares de STP	25 m	Par trançado blindado
1000Base-T	4 pares de UTP	100 m	UTP padrão da categoria 5

[F]Figura 4.23

[FL] O cabeamento da Ethernet de gigabit

São permitidos três diâmetros de fibra: 10, 50 e 62,5 micra. O primeiro se destina ao modo único e os dois últimos ao multimodo. Contudo, nem todas as seis combinações são permitidas, e a distância máxima depende da combinação usada. Os números dados na Figura 4.23 se referem ao melhor caso. Em particular, a distância de 5000 metros só pode ser alcançada com lasers de 1,3 mícron operando sobre fibra de 10 micra em modo único, mas essa é a melhor opção para backbones de campus, e espera-se que ela se torne popular, apesar de ser a opção mais cara.

A opção 1000Base-CX utiliza cabos curtos de cobre blindado. O problema é que ela está competindo com fibra de alto desempenho no limite superior e com o UTP de baixo custo no limite inferior. É improvável que ela seja muito usada, ou mesmo que seja usada.

A última opção emprega grupos de quatro fios UTP da categoria 5 reunidos.

Como grande parte dessa fiação já está instalada, é provável que ela forme a Ethernet de gigabit popular.

A Ethernet de gigabit utiliza novas regras de codificação nas fibras. A codificação de Manchester a 1 Gbps exigiria um sinal de 2 Gbauds, considerada algo muito difícil e também um grande desperdício de largura de banda. Em vez disso, foi escolhido um novo esquema, chamado **8B/10B**, baseado em um canal de fibra. Cada byte de 8 bits é codificado na fibra como 10 bits, daí o nome 8B/10B. Tendo em vista que existem 1024 palavras de código de saída possíveis para cada byte de entrada, houve uma certa tolerância na escolha das palavras de código que seriam permitidas. As duas regras a seguir foram usadas na escolha:

1. Nenhuma palavra de código pode ter mais de quatro bits idênticos em seqüência.
2. Nenhuma palavra de código pode ter mais de seis valores 0, ou seis valores 1.

Essas escolhas foram feitas para manter transições suficientes no fluxo, a fim de assegurar que o receptor permanecerá sincronizado com o transmissor e também para manter o número de valores 0 e 1 na fibra o mais próximo possível da igualdade. Além disso, muitos bytes de entrada têm duas palavras de código possíveis atribuídas a eles. Quando o codificador tem a opção de selecionar palavras de código, ele sempre escolhe a palavra de código que o leva na direção da igualdade entre o número de valores 0 e o de valores 1 transmitidos até o momento. Essa ênfase em equilibrar os valores 0 e 1 é necessária para manter o componente CC do sinal tão baixo quanto possível, a fim de permitir que ele passe por transformadores sem ser modificado. Embora os cientistas da computação não gostem que as propriedades dos transformadores determinem seus esquemas de codificação, às vezes a vida é assim.

As redes Ethernet de gigabit que utilizam 1000Base-T empregam um esquema de codificação diferente, pois a verificação de dados no fio de cobre em um 1 ns é

muito difícil. Essa solução utiliza quatro pares trançados da categoria 5 para permitir a transmisssão de quatro símbolos em paralelo. Cada símbolo é codificado com o uso de um entre cinco níveis de voltagem. Esse esquema permite que um único símbolo codifique 00, 01, 10, 11 ou um valor especial para fins de controle. Desse modo, existem 2 bits de dados por par trançado ou 8 bits de dados por ciclo de clock. O clock funciona a 125 MHz, permitindo operação a 1 Gbps. A razão para permitir cinco níveis de voltagem em vez de quatro é ter combinações de sobra para fins de enquadramento e controle.

Uma velocidade de 1 Gbps é bastante alta. Por exemplo, se um receptor estiver ocupado com alguma outra tarefa, mesmo durante 1 ms, e não esvaziar o buffer de entrada em alguma linha, poderão se acumular até 1953 quadros nesse intervalo de 1 ms. Além disso, quando um computador em uma Ethernet de gigabit estiver transmitindo dados pela linha a um computador em uma Ethernet clássica, serão muito prováveis sobrecargas no buffer. Como conseqüência dessas duas observações, a Ethernet de gigabit admite controle de fluxo (como a Fast Ethernet, embora os dois padrões sejam diferentes).

O controle de fluxo consiste na transmissão de um quadro de controle especial por uma extremidade para a outra, informando que a extremidade receptora deve fazer uma pausa durante algum período de tempo predeterminado. Os quadros de controle são quadros Ethernet normais contendo um tipo de 0x8808. Os dois primeiros bytes do campo de dados fornecem o comando; os bytes seguintes fornecem os parâmetros, se houver. Para controle de fluxo, são usados quadros PAUSE, com o parâmetro informando quanto tempo deve durar a pausa, nas unidades do tempo mínimo de quadro. Para a Ethernet de gigabit, a unidade de tempo é 512 ns, permitindo pausas de até 33,6 ms.

Assim que a Ethernet de gigabit foi padronizada, o comitê 802 ficou entediado e resolveu voltar a trabalhar. O IEEE solicitou que eles começassem a atuar na

Ethernet de 10 gigabits. Após uma busca árdua por uma letra para acompanhar a letra z, o comitê abandonou essa abordagem e passou a usar sufixos de duas letras. Seus participantes começaram a trabalhar e o padrão foi aprovado pelo IEEE no ano de 2002, como o padrão 802.3ae. Será que a Ethernet de 100 gigabits está muito longe?

[T3] 4.3.9 O padrão IEEE 802.2: LLC (Logical Link Control)

Talvez seja a hora de recuar um pouco e comparar o que aprendemos neste capítulo com o que estudamos no capítulo anterior. No Capítulo 3, vimos como duas máquinas poderiam se comunicar de modo confiável sobre uma linha não confiável, usando diversos protocolos de enlace de dados. Esses protocolos ofereciam recursos de controle de erros (com o uso de confirmações) e controle de fluxo (com o uso de uma janela deslizante).

Por outro lado, neste capítulo, ainda não dissemos nada sobre comunicação confiável. Tudo que a Ethernet e os outros protocolos 802 oferecem é um serviço de datagrama de boa qualidade. Às vezes, esse serviço é adequado. Por exemplo, para o transporte de pacotes IP não há garantias, nem se espera que elas existam. Um pacote IP só pode ser inserido em um campo de carga útil 802 e transmitido em seguida. Se ele se perder, não há nada que possa ser feito.

Todavia, também existem sistemas em que um protocolo de enlace de dados com controle de fluxo e controle de erros se faz necessário. O IEEE definiu um protocolo que pode funcionar sobre a Ethernet e sobre os outros protocolos 802. Além disso, esse protocolo — chamado **LLC (Logical Link Control — controle de enlace lógico)** — oculta as diferenças entre os diversos tipos de redes 802, fornecendo um único formato e uma única interface com a camada da rede. Esse formato, a interface e o protocolo se baseiam principalmente no modelo HDLC que estudamos no Capítulo 3. O LLC forma a metade superior da camada de

enlace de dados, com a subcamada MAC abaixo dele, como mostra a Figura 4.24. O uso mais comum do LLC é descrito a seguir. A camada de rede da máquina de transmissão repassa um pacote para o LLC, usando as primitivas de acesso do LLC. A subcamada LLC acrescenta o cabeçalho LLC que contém números de seqüência e de confirmação. A estrutura resultante é então inserida no campo de carga útil de um quadro 802 e, em seguida, é transmitida. No receptor, ocorre o processo inverso.

O LLC fornece três opções de serviço: serviço de datagrama não confiável, serviço de datagrama com confirmação e serviço confiável orientado a conexões. O cabeçalho do LLC contém três campos: um ponto de acesso de destino, um ponto de acesso de origem e um campo de controle. Os pontos de acesso informa de que processo o quadro veio e onde ele deve ser entregue, substituindo o campo *Tipo* do DIX. O campo de controle contém números de seqüência e confirmação, em estilo muito semelhante ao do HDLC (ver Figura 3.24), mas não idêntico a ele. Esses campos são usados principalmente quando é necessária uma conexão confiável no nível de enlace de dados e, nesse caso, seriam usados protocolos semelhantes aos que descrevemos no Capítulo 3. No caso da Internet, é suficiente tentar fazer o melhor possível para entregar pacotes IP; assim, não é necessária nenhuma confirmação no nível de LLC.

[arte: imagem original da p. 291]

[Dísticos]

[1] Camada de rede

[2] Camada de enlace de dados

[3] LLC

MAC

[4] Camada física

(a)

[5] Pacote

[6] LLC Pacote

[7] MAC LLC Pacote MAC

[8] Rede

(b)

[F]Figura 4.24

[FL] (a) Posição do LLC. (b) Formatos de protocolos

[T3] 4.3.10 Retrospectiva da Ethernet

A Ethernet existe há mais de 20 anos e não tem concorrentes sérios; portanto, é provável que continue no mercado por muitos anos ainda. Poucas arquiteturas de CPUs, sistemas operacionais ou linguagens de programação têm se mantido na liderança por mais de duas décadas. Sem dúvida, a Ethernet tem algumas características que justificam essa liderança. Quais são elas?

Provavelmente a principal razão para sua longevidade seja o fato de que a Ethernet é simples e flexível. Na prática, simples se traduz como confiável, de baixo custo e de fácil manutenção. Depois que as derivações foram substituídas por conectores BNC, as falhas se tornaram extremamente raras. As pessoas hesitam em substituir algo que funciona bem o tempo todo, em especial quando sabem que uma quantidade terrível de itens da indústria de informática funciona muito mal. Muitas das chamadas "atualizações" são bem piores que as versões substituídas por elas.

Simplicidade também se traduz em economia. O cabeamento Ethernet fino e a fiação de par trançado têm custo relativamente baixo. As placas de interface também têm baixo custo. Somente quando os hubs e switches foram introduzidos, surgiu a necessidade de investimentos significativos mas, na época em que eles entraram em cena, a Ethernet já estava bem estabelecida.

A Ethernet é de fácil manutenção. Não existe nenhum software para instalar (além dos drivers) e não há nenhuma tabela de configuração para gerenciar (e errar). Além disso, a inclusão de novos hosts é simples: basta conectá-los.

Outro ponto importante é que a Ethernet é capaz de interoperar facilmente com o TCP/IP, que se tornou dominante. O IP é um protocolo sem conexões, e portanto se ajusta perfeitamente à Ethernet, que também é sem conexões. O IP não tem a mesma facilidade para se ajustar ao ATM, que é orientado a conexões. Essa falta de compatibilidade definitivamente diminui as chances de sucesso do ATM.

Por fim, a Ethernet foi capaz de evoluir em certos aspectos cruciais. As velocidades aumentaram várias ordens de magnitude, e os hubs e switches foram introduzidos, mas essas mudanças não exigiram alterações no software. Quando um vendedor de redes mostra uma grande instalação e diz: "Tenho esta nova e fantástica rede para você. Basta se desfazer de todo seu hardware e reescrever todo o seu software", ele tem um problema. Ao serem lançados, o FDDI, o Fibre Channel e o ATM eram mais rápidos que a Ethernet, mas eram incompatíveis com a Ethernet, muito mais complexos e mais difíceis de gerenciar. Eventualmente, a Ethernet os alcançou em termos de velocidade, e assim eles não tiveram nenhuma outra vantagem a oferecer e logo desapareceram, exceto pelo uso do ATM dentro do núcleo do sistema telefônico.

[T2] 4.4 LANs sem fios

Embora a Ethernet seja amplamente utilizada, ela está prestes a enfrentar alguma concorrência. As LANs sem fios estão cada vez mais populares e um número crescente de edifícios de escritórios, aeroportos e outros lugares públicos estão sendo equipados com elas. As LANs sem fios podem operar em duas configurações, como vimos na Figura 1.35: com e sem uma estação base. Conseqüentemente, o padrão de LAN 802.11 leva em conta esse fato e prevê

ambas as organizações, conforme veremos em breve.

Vimos algumas informações básicas sobre o padrão 802.11 na Seção 1.5.4. Agora, vamos examinar mais de perto a tecnologia. Nas próximas seções, estudaremos a pilha de protocolos, as técnicas de transmissão de rádio da camada física, o protocolo da subcamada MAC, a estrutura de quadro e os serviços. Para obter mais informações sobre o 802.11, consulte (Crow *et al.*, 1997; Geier, 2002; Heegard *et al.*, 2001; Kapp, 2002; O'Hara e Petrick, 1999; e Severance, 1999). Para conhecer os detalhes mais profundos, consulte o próprio padrão 802.11 publicado.

[T3] 4.4.1 802.11: a pilha de protocolos

Os protocolos usados por todas as variantes do 802, inclusive a Ethernet, têm certas características comuns em sua estrutura. Uma visão parcial da pilha de protocolos do 802.11 é dada na Figura 4.25. A camada física corresponde muito bem à camada física do modelo OSI, mas a camada de enlace de dados em todos os protocolos 802 se divide em duas ou mais subcamadas. No 802.11, a subcamada MAC (Medium Access Control) determina como o canal é alocado, isto é, quem terá a oportunidade de transmitir em seguida. Acima dela, encontra-se a subcamada LLC (Logical Link Control), cujo trabalho é ocultar as diferenças entre as diversas variações do 802 e torná-las indistinguíveis no que se refere à camada de rede. Estudamos a subcamada LLC quando examinamos a Ethernet em uma seção anterior deste capítulo e não repetiremos esse assunto aqui.

O padrão 802.11 de 1997 especifica três técnicas de transmissão permitidas na camada física. O método de infravermelho utiliza quase a mesma tecnologia que os controles remotos dos televisores. Os outros dois métodos empregam rádio de alcance limitado, utilizando técnicas chamadas FHSS e DSSS. Ambas utilizam uma parte do espectro que não exige licenciamento (a banda ISM de 2,4 GHz). Os

dispositivos de abertura de portas de garagem controlada por rádio também empregam essa parte do espectro, e assim seu notebook pode acabar competindo com a porta da sua garagem. Os telefones sem fios e os fornos de microondas também utilizam essa banda. Todas essas técnicas operam a 1 ou 2 Mbps e com baixa potência, suficiente para evitar muitos conflitos. Em 1999, foram apresentadas duas novas técnicas para alcançar maior largura de banda. Essas técnicas são chamadas OFDM e HR-DSSS. Elas operam em até 54 Mbps e 11 Mbps, respectivamente. Em 2001, uma segunda modulação de OFDM foi introduzida, mas em uma banda de freqüência diferente da primeira. Agora, vamos examinar cada uma delas em linhas gerais. Tecnicamente, elas pertencem à camada física e deveriam ter sido examinadas no Capítulo 2; porém, como estão estritamente relacionadas às LANs em geral e à subcamada MAC do 802.11, preferimos tratá-las aqui.

[arte: imagem original da p. 293]

[Dísticos]

[1]Camadas superiores

[2]Controle de enlace lógico

[3]Camada de enlace de dados

[4]Camada física

[5]802.11 Infravermelho 802.11 FHSS 802.11 DSSS 802.11a OFDM 802.11b HR-DSSS 802.11g OFDM

[6]Subcamada MAC

[F]Figura 4.25

[FL] Parte da pilha de protocolos do 802.11

[T3] 4.4.2 802.11: a camada física

Cada uma das cinco técnicas de transmissão permitidas torna possível enviar um

quadro MAC de uma estação para outra. Contudo, elas diferem na tecnologia usada e nas velocidades que podem ser alcançadas. Uma descrição detalhada dessas tecnologias está muito além do escopo deste livro, mas algumas palavras sobre cada uma, juntamente com algumas palavras-chaves, podem fornecer aos leitores interessados material para pesquisar mais informações na Internet ou em outras fontes.

A opção de infravermelho usa transmissão difusa (isto é, não linear) a 0,85 ou 0,95 mícron. São permitidas duas velocidades: 1 Mbps e 2 Mbps. A 1 Mbps, é usado um esquema de codificação no qual um grupo de 4 bits é codificado como uma palavra de código de 16 bits, contendo quinze bits 0 e um único bit 1, empregando o que chamamos **código de Gray.** Esse código se caracteriza pela propriedade de um pequeno erro na sincronização resultar em apenas um erro de bit na saída. A 2 Mbps, a codificação ocupa 2 bits e produz uma palavra de código de 4 bits, também com apenas um bit 1, que pode ser 0001, 0010, 0100 ou 1000. Os sinais de infravermelho não podem atravessar paredes; assim, células situadas em salas diferentes ficam bem isoladas umas das outras. Apesar disso, devido à baixa largura de banda (e ao fato de que a luz solar altera os sinais de infravermelho), essa não é uma opção popular.

O **FHSS (Frequency Hopping Spread Spectrum — espectro de dispersão de saltos de freqüência)** utiliza 79 canais, cada um com 1 MHz de largura, começando na extremidade baixa da banda ISM de 2,4 GHz. Um gerador de números pseudo-aleatórios é usado para produzir a seqüência de freqüências dos saltos. Desde que todas as estações utilizem a mesma semente para o gerador de números pseudo-aleatórios e permaneçam sincronizadas, elas saltarão para as mesmas freqüências simultaneamente. O período de tempo gasto em cada freqüência, o **@@@tempo de parada**, é um parâmetro ajustável, mas deve ser menor que 400 ms. A randomização do FHSS fornece um modo razoável de alocar espectro na

banda ISM não regulamentada. Ela também fornece alguma segurança, pois um intruso que não conhecer a seqüência de saltos ou o tempo de parada não poderá espionar as transmissões. Em distâncias mais longas, o esmaecimento de vários caminhos pode ser um problema, e o FHSS oferece boa resistência a ele. O FHSS também é relativamente insensível à interferência de rádio, o que o torna popular para enlaces entre edifícios. Sua principal desvantagem é a baixa largura de banda.

O terceiro método de modulação, o **DSSS (Direct Sequence Spread Spectrum — espectro de dispersão de seqüência direta)**, também é restrito a 1 ou 2 Mbps. O esquema usado tem algumas semelhanças em relação ao sistema CDMA que examinamos na Seção 2.6.2, mas difere deste último em outros aspectos. Cada bit é transmitido como 11 chips, usando o que se denomina **seqüência de Barker.** Ele utiliza modulação por deslocamento de fase a 1 Mbaud, transmitindo 1 bit por baud quando opera a 1 Mbps e 2 bits por baud quando opera a 2 Mbps. Durante anos, a FCC exigiu que todo o equipamento de comunicações sem fios operasse nas bandas ISM nos Estados Unidos, a fim de utilizar o espectro de dispersão; porém, em maio de 2002, essa regra foi abandonada após surgirem novas tecnologias.

A primeira das LANs sem fios de alta velocidade, a LAN **802.11a**, utiliza **OFDM (Orthogonal Frequency Division Multiplexing — multiplexação ortogonal por divisão de freqüência)** para transmitir até 54 Mbps na banda ISM mais larga, de 5 GHz. Como sugere o termo FDM, são usadas diferentes freqüências — 52 delas, sendo 48 para dados e 4 para sincronização — de modo semelhante ao ADSL. Tendo em vista que as transmissões estão presentes em várias freqüências ao mesmo tempo, essa técnica é considerada uma forma de espectro de dispersão, mas diferente do CDMA e do FHSS. A divisão do sinal em muitas bandas estreitas tem algumas vantagens fundamentais em relação ao uso de uma única banda

larga, incluindo melhor imunidade à interferência de banda estreita, e a possibilidade de usar bandas não contíguas. É usado um sistema de codificação complexo, baseado na modulação por deslocamento de fase, a fim de alcançar velocidades de até 18 Mbps e, na QAM, velocidades acima dessas. A 54 Mbps, 216 bits de dados são codificados em símbolos de 288 bits. Parte da motivação para a OFDM é a compatibilidade com o sistema europeu HiperLAN/2 (Doufexi *et al.*, 2002). A técnica tem boa eficiência de espectro em termos de bits/Hz e boa imunidade ao esmaecimento de vários caminhos.

Em seguida, vamos ao **HR-DSSS (High Rate Direct Sequence Spread Spectrum — espectro de dispersão de seqüência direta de alta velocidade)**, outra técnica de espectro de dispersão, que utiliza 11 milhões de chips/s para alcançar 11 Mbps na banda de 2,4 GHz. Ela é chamada **802.11b**, mas não é uma continuação do 802.11a. De fato, seu padrão foi aprovado e chegou primeiro ao mercado. As taxas de dados admitidas pelo 802.11b são 1, 2, 5,5 e 11 Mbps. As duas taxas mais baixas funcionam a 1 Mbaud, com 1 e 2 bits por baud, respectivamente, usando a modulação por deslocamento de fase (por compatibilidade com o DSSS). As duas taxas mais rápidas funcionam a 1,375 Mbaud, com 4 e 8 bits por baud, respectivamente, usando códigos de **Walsh/Hadamard**. A taxa de dados pode ser adaptada dinamicamente durante a operação para alcançar a velocidade ótima possível sob as condições atuais de carga e ruído. Na prática, a velocidade de operação do 802.11b é quase sempre igual a 11 Mbps. Embora o 802.11b seja mais lento que o 802.11a, seu alcance é cerca de 7 vezes maior, o que é mais importante em muitas situações.

Uma versão aperfeiçoada do 802.11b, o **802.11g**, foi aprovada pelo IEEE em novembro de 2001, depois de muitas disputas políticas sobre qual tecnologia patenteada seria usada. Ele utiliza o método de modulação OFDM do 802.11ª, mas opera na banda ISM estreita de 2,4 GHz, juntamente com o 802.11b. Em

tese, ele pode operar em até 54 Mbps. Ainda não está claro se essa velocidade será alcançada na prática. Isso significa que o comitê 802.11 produziu três diferentes LANs sem fios de alta velocidade: 802.11a, 802.11b e 802.11g (sem mencionarmos três LANs sem fios de baixa velocidade). Seria legítimo perguntar se essa é uma boa medida para um comitê de padrões. Talvez três seja seu número da sorte.

[T3] 4.4.3 802.11: o protocolo da subcamada MAC

Agora, vamos retornar dos domínios da engenharia elétrica para os da ciência de computação. O protocolo da subcamada MAC do 802.11 é bastante diferente do protocolo da Ethernet, devido à complexidade inerente do ambiente sem fio, em comparação com o de um sistema fisicamente conectado. Com a Ethernet, uma estação só precisa esperar até o éter ficar inativo e começar a transmitir. Se não receber de volta uma rajada de ruído dentro dos primeiros 64 bytes, é quase certo que o quadro tenha sido entregue corretamente. No caso das LANs sem fios, essa situação não ocorre.

Para começar, existe o problema da estação oculta mencionado antes e ilustrado mais uma vez na Figura 4.26(a). Tendo em vista que nem todas as estações estão dentro do alcance de rádio umas das outras, as transmissões realizadas em uma parte de uma célula podem não ser recebidas em outros lugares na mesma célula. Nesse exemplo, a estação *C* está transmitindo para a estação *B*. Se *A* escutar o canal, não ouvirá nada e concluirá erradamente que agora pode iniciar a transmissão para *B*.

Além disso, existe o problema inverso, o problema da estação exposta, ilustrado na Figura 4.26(b). Agora *B* quer transmitir para *C*, e portanto escuta o canal. Quando ouve uma transmissão, a estação *B* conclui erradamente que não pode transmitir para *C*, embora *A* talvez esteja transmitindo para *D* (não mostrada).

Além disso, a maioria dos rádios é half-duplex, significando que eles não podem transmitir e ouvir rajadas de ruído ao mesmo tempo em uma única freqüência. Como resultado desses problemas, o 802.11 não utiliza o CSMA/CD, como faz o padrão Ethernet.

[arte: imagem original da p. 296]

[Dísticos]

[1]A quer enviar para B, mas não pode ouvir que B está ocupada

[2]B quer enviar para C, mas pensa erradamente que a transmissão falhará

[3]Alcance do rádio de C

A B C

C está transmitindo

(a)

[4]Alcance do rádio de A

A B C

A está transmitindo

(b)

[F]Figura 4.26

[FL] (a) O problema da estação oculta. (b) O problema da estação exposta

Para lidar com esse problema, o 802.11 admite dois modos de operação. O primeiro, chamado **DCF (Distributed Coordination Function — função de coordenação distribuída)**, não usa nenhuma espécie de controle central (nesse aspecto, ele é semelhante ao padrão Ethernet). O outro, chamado **PCF (Point Coordination Function — função de coordenação de ponto)**, utiliza a estação base para controlar toda a atividade em sua célula. Todas as implementações devem aceitar DCF, mas PCF é opcional. Agora, vamos descrever esses dois modos.

Quando se emprega o modo DCF, o 802.11 utiliza um protocolo chamado **CSMA/CA (CSMA with Collision Avoidance — CSMA com abstenção de colisão).** Nesse protocolo, são usadas tanto a detecção do canal físico quanto a do canal virtual. O CSMA/CA admite dois métodos de operação. No primeiro método, quando uma estação quer transmitir, ela escuta o canal. Se ele estiver ocioso, a estação simplesmente começará a transmitir. Ela não escuta o canal enquanto está transmitindo, mas emite seu quadro inteiro, que pode muito bem ser destruído no receptor devido à interferência. Se o canal estiver ocupado, a transmissão será adiada até o canal ficar inativo, e então a estação começará a transmitir. Se ocorrer uma colisão, as estações que colidirem terão de esperar um tempo aleatório, usando o algoritmo de recuo binário exponencial das redes Ethernet, e então tentarão novamente mais tarde.

O outro modo de operação do CSMA/CA se baseia no MACAW e emprega a detecção de canal virtual, como ilustra a Figura 4.27. Nesse exemplo, *A* quer transmitir para *B*. *C* é uma estação dentro do alcance de *A* (e possivelmente dentro do alcance de *B*, mas isso não importa). *D* é uma estação dentro do alcance de *B*, mas não dentro do alcance de *A*.

O protocolo começa quando *A* decide transmitir dados para *B*. Ela inicia a transmissão enviando um quadro RTS para *B*, a fim de solicitar permissão para enviar um quadro. Quando recebe essa solicitação, *B* pode decidir conceder a permissão e, nesse caso, envia de volta um quadro CTS. Após a recepção do CTS, *A* envia seu quadro e inicia um timer ACK. Ao receber corretamente o quadro de dados, *B* responde com um quadro ACK, concluindo a troca de quadros. Se o timer ACK de *A* expirar antes do quadro ACK voltar a ele, o protocolo inteiro será executado novamente.

[arte: imagem original da p. 297]

[Dísticos]

[1]A RTS Dados

[2]B CTS ACK

[3]C NAV

[4]D NAV

[5]Tempo

[F]Figura 4.27

[FL] O uso da detecção de canal virtual com o CSMA/CA

Agora, vamos considerar essa troca sob os pontos de vista de *C* e *D*. *C* está dentro do alcance de *A*, e então pode receber o quadro RTS. Se o fizer, *C* perceberá que alguém vai transmitir dados em breve e assim, para o bem de todos, desiste de transmitir qualquer coisa até a troca ser concluída. A partir das informações fornecidas na solicitação RTS, ela pode avaliar quanto tempo a seqüência irá demorar, incluindo o ACK final, e assim reivindica uma espécie de canal virtual ocupado por ela própria, indicado por **NAV (Network Allocation Vector — vetor de alocação de rede)** na Figura 4.27. *D* não escuta o RTS, mas escuta o CTS, e assim também reivindica o sinal *NAV* para ela própria. Observe que os sinais *NAV* não são transmitidos; eles são apenas lembretes internos de que a estação deve se manter inativa por um determinado período de tempo.

Em contraste com as redes fisicamente conectadas, as redes sem fios são ruidosas e pouco confiáveis, em grande parte como os fornos de microondas, que também utilizam as bandas ISM não licenciadas. Em conseqüência disso, a probabilidade de um quadro trafegar por elas com sucesso diminui com o comprimento do quadro. Se a probabilidade de ocorrer um erro em qualquer bit é p, então a probabilidade de um quadro de n bits ser recebido de forma inteiramente correta é $(1 - p)^n$. Por exemplo, para $p = 10^{-4}$, a probabilidade de receber um quadro Ethernet completo (12.144 bits) sem erros é menor que 30%.

Se $p = 10^{-5}$, aproximadamente 1 quadro em 9 será danificado. Ainda que $p = 10^{-6}$, haverá danos em mais de 1% dos quadros, o que significa quase uma dezena de quadros danificados por segundo, ou mais que isso se forem usados quadros mais curtos que o comprimento máximo. Em resumo, se um quadro for longo demais, ele terá bem pouca chance de chegar sem danos e é provável que tenha de ser retransmitido.

Para lidar com o problema de canais ruidosos, o 802.11 permite que os quadros sejam fragmentados em partes menores, cada uma com seu próprio total de verificação. Os fragmentos são numerados individualmente e confirmados com o uso de um protocolo do tipo stop-and-wait (isto é, o transmissor não pode enviar o fragmento $k + 1$ enquanto não receber a confirmação do fragmento k). Depois que um canal é adquirido com o uso de RTS e CTS, vários fragmentos podem ser enviados em seqüência, como mostra a Figura 4.28. A seqüência de fragmentos é chamada **rajada de fragmentos**.

[arte: imagem original da p. 298]

[Dísticos]

[1] Rajada de fragmentos

A RTS Frag 1 Frag 2 Frag 3

[2]B CTS ACK ACK ACK

[3]C NAV

[4]D NAV

[5] Tempo

[F]Figura 4.28

[FL] Uma rajada de fragmentos

A fragmentação aumenta o throughput, restringindo as retransmissões aos fragmentos defeituosos, em vez de retransmitir o quadro inteiro. O tamanho do

fragmento não é fixado pelo padrão, mas é um parâmetro de cada célula e pode ser ajustado pela estação base. O mecanismo NAV mantém outras estações inativas apenas até a próxima confirmação, mas outro mecanismo (descrito a seguir) é usado para permitir que uma rajada de fragmentos inteira seja enviada sem interferência.

Toda a discussão anterior se aplica ao modo DCF do 802.11. Nesse modo, não existe nenhum controle central, e as estações concorrem pelo tempo no ar, da mesma forma que concorrem no caso da Ethernet. O outro modo permitido é o PCF, no qual a estação base efetua o polling das outras estações, perguntando se elas têm algum quadro a enviar. Tendo em vista que a ordem de transmissão é totalmente controlada pela estação base em modo PCF, não ocorre nenhuma colisão. O padrão prescreve o mecanismo de polling, mas não a freqüência de polling, a ordem do polling, ou mesmo se todas as estações precisam receber um atendimento idêntico.

O mecanismo básico consiste na difusão periódica pela estação base de um **quadro de baliza** (de 10 a 100 vezes por segundo). O quadro de baliza contém parâmetros do sistema, como seqüências de saltos (hops) e tempos de parada (para o FHSS), sincronização do clock etc. Ele também convida novas estações a se inscreverem no serviço de polling. Depois que uma estação se inscreve para receber o serviço de polling a uma certa taxa, ela tem a garantia efetiva de uma certa fração da largura de banda, tornando possível assim oferecer garantias de qualidade de serviço.

A duração da bateria é sempre um problema nos dispositivos móveis sem fios, e assim o 802.11 dedica atenção à questão do gerenciamento de energia. Em particular, a estação base pode orientar uma estação móvel a entrar no estado de espera até ser despertada explicitamente pela estação base ou pelo usuário. Contudo, tendo orientado uma estação para ficar inativa, isso significa que a

estação base tem a responsabilidade de armazenar no buffer quaisquer quadros dirigidos a ela, enquanto a estação móvel estiver inativa. Esses quadros podem ser reunidos mais tarde.

PCF e DCF podem coexistir dentro de uma única célula. À primeira vista, pode parecer impossível ter o controle central e o controle distribuído operando ao mesmo tempo, mas o 802.11 fornece um meio para atingir esse objetivo. Ele funciona definindo com todo cuidado o intervalo de tempo entre quadros. Depois que um quadro é enviado, é exigido um certo período de tempo de inatividade, antes que qualquer estação possa enviar um quadro. São definidos quatro intervalos distintos, cada um correspondendo a uma finalidade específica. Os quatro intervalos estão representados na Figura 4.29.

[arte: imagem original da p. 299]

[Dísticos]

[1]Quadro de controle ou próximo fragmento pode ser enviado aqui

[2]SIFS Quadros PCF podem ser enviados aqui

[3]PIFS Quadros DCF podem ser enviados aqui

[4]A recuperação de quadros defeituosos é feita aqui

[5]DIFS

EIFS

[6]ACK

Tempo

[F]Figura 4.29

[FL] Espaçamento entre quadros no 802.11

O menor intervalo é o **SIFS (Short InterFrame Spacing — espaçamento curto entre quadros)**. Ele é usado para permitir que as partes de um único diálogo tenham a chance de transmitir primeiro. Isso inclui a permissão para que o receptor envie

um CTS, a fim de responder a um RTS, deixando o receptor enviar um ACK relativo a um fragmento ou a todo o quadro de dados, e deixando o transmissor de uma rajada de fragmentos transmitir o próximo fragmento sem ter de enviar um RTS novamente.

Sempre existe exatamente uma estação habilitada a responder após um intervalo SIFS. Se ela deixar de fazer uso de sua chance e decorrer um tempo **PIFS (PCF InterFrame Spacing — espaçamento entre quadros PCF)**, a estação base poderá enviar um quadro de baliza ou um quadro de polling. Esse mecanismo permite a uma estação transmitir um quadro de dados ou uma seqüência de fragmentos para encerrar seu quadro sem a interferência de qualquer outro, mas oferece à estação base a chance de se apoderar do canal quando o transmissor anterior terminar, sem ter de competir com usuários ávidos.

Se a estação base não tiver nada a transmitir e decorrer um tempo **DIFS (DCF InterFrame Spacing — espaçamento entre quadros DCF)**, qualquer estação poderá tentar adquirir a posse do canal para enviar um novo quadro. As regras habituais de disputa se aplicam, e o recuo binário exponencial pode ser necessário, se ocorrer uma colisão.

O último intervalo de tempo, **EIFS (Extended InterFrame Spacing — espaçamento estendido entre quadros)**, só é usado por uma estação que tenha acabado de receber um quadro defeituoso ou desconhecido, a fim de informar sobre a presença do quadro defeituoso. A idéia é dar a esse evento a prioridade mais baixa pois, como o receptor talvez não tenha nenhuma idéia do que está acontecendo, ele deve esperar um tempo significativo para evitar interferir com um diálogo em andamento entre duas estações.

[T3] 4.4.4 802.11: estrutura de quadro

O padrão 802.11 define três diferentes classes de quadros em trânsito: dados,

controle e gerenciamento. Cada um deles tem um cabeçalho com uma variedade de campos usados na subcamada MAC. Além disso, existem alguns cabeçalhos usados pela camada física, mas eles lidam principalmente com as técnicas de modulação empregadas e, portanto, não os discutiremos aqui.

O formato do quadro de dados é mostrado na Figura 4.30. Primeiro vem o campo *Controle de quadro*. Ele próprio tem 11 subcampos. O primeiro desses subcampos denomina-se *Versão do protocolo*, que permite a operação de duas versões do protocolo ao mesmo tempo na mesma célula. Depois, temos os campos *Tipo* (dados, controle ou gerenciamento) e *Subtipo* (por exemplo, RTS ou CTS). Os bits *Para DS* e *De DS* indicam se o quadro está indo ou vindo do sistema de distribuição entre células (por exemplo, Ethernet). O bit *MF* significa que haverá mais fragmentos. O bit *Repetir* indica uma retransmissão de um quadro enviado anteriormente. O bit *Gerenciamento de energia* é usado pela estação base para deixar o receptor em estado de espera ou retirá-lo do estado de espera. O bit *Mais* indica que o transmissor tem quadros adicionais para o receptor. O bit *W* especifica que o corpo de quadro foi criptografado com o algoritmo **WEP (Wired Equivalent Privacy — privacidade equivalente quando fisicamente conectado).** Por fim, o bit *O* informa ao receptor que uma seqüência de quadros com esse bit tem de ser processada estritamente em ordem.

[arte: imagem original da p. 300]

[Dísticos]

[1]Bytes 2 2 6 6 6 2 6 0-2312 4

[2]Controle de quadro Duração Endereço 1 Endereço 2 Endereço 3 Seq. Endereço 4 Dados Total de verificação

[3]Bits 2 2 4 1 1 1 1 1 1 1 1

[4]Versão Tipo Subtipo Para DS De DS MF Repetir Potência Mais W O Controle de quadro

[F]Figura 4.30

[FL] O quadro de dados do 802.11

O segundo campo do quadro de dados, o campo *Duração*, informa por quanto tempo o quadro e sua confirmação ocuparão o canal. Esse campo também está presente nos quadros de controle e representa a forma como outras estações administram o mecanismo NAV. O cabeçalho de quadro contém quatro endereços, todos em formato padrão IEEE 802. É óbvio que a origem e o destino são necessários, mas quais são os outros dois? Lembre-se de que os quadros podem entrar ou sair de uma célula por meio de uma estação base. Os outros dois endereços são utilizados pelas estações base de origem e destino para tráfego entre células.

O campo *Seqüência* permite que os fragmentos sejam numerados. Dos 16 bits disponíveis, 12 identificam o quadro e 4 identificam o fragmento. O campo *Dados* contém a carga útil de até 2312 bytes, e é seguido pelo campo habitual *Total de verificação*.

Os quadros de gerenciamento têm um formato semelhante ao dos quadros de dados, exceto por não terem um dos endereços da estação base, porque os quadros de gerenciamento estão restritos a uma única célula. Os quadros de controle são ainda mais curtos, tendo apenas um ou dois endereços, nenhum campo *Dados* e nenhum campo *Seqüência*. Nesse caso, a informação importante está no campo *Subtipo*, em geral RTS, CTS ou ACK.

[T3] 4.4.5 Serviços

O padrão 802.11 estabelece que cada LAN sem fio compatível deve fornecer nove serviços. Esses serviços estão divididos em duas categorias: cinco serviços de distribuição e quatro serviços da estação. Os serviços de distribuição se

relacionam ao gerenciamento da associação a células e à interação com estações situadas fora da célula. Em contraste, os serviços da estação se relacionam à atividade dentro de uma única célula.

Os cinco serviços de distribuição são fornecidos pelas estações base e lidam com a mobilidade das estações à medida que elas entram e saem das células, conectando-se e desconectando-se das estações base. Esses serviços são apresentados a seguir.

1. **Associação.** Esse serviço é usado pelas estações móveis para conectá-las às estações base. Em geral, ele é usado imediatamente após uma estação se deslocar dentro do alcance de rádio da estação base. Ao chegar, ela anuncia sua identidade e seus recursos. Os recursos incluem as taxas de dados admitidas, a necessidade de serviços PCF (isto é, polling) e requisitos de gerenciamento da energia. A estação base pode aceitar ou rejeitar a estação móvel. Se for aceita, a estação móvel terá de se autenticar.

2. **Desassociação.** A estação móvel ou a estação base pode se desassociar, interrompendo assim o relacionamento. Uma estação deve usar esse serviço antes de se desligar ou sair, mas a estação base também pode usá-lo antes de se desativar para manutenção.

3. **Reassociação.** Uma estação pode mudar sua estação base preferida usando esse serviço. Esse recurso é útil para estações móveis que se deslocam de uma célula para outra. Se for usado corretamente, não haverá perda de dados em conseqüência da transferência (handover). (Porém, o 802.11, como o padrão Ethernet, é apenas um serviço que faz o melhor possível.)

4. **Distribuição.** Esse serviço determina como rotear quadros enviados à estação base. Se o destino for local para a estação base, os quadros poderão ser enviados diretamente pelo ar. Caso contrário, eles terão de ser encaminhados pela rede fisicamente conectada.

5. **Integração.** Se um quadro precisar ser enviado por meio de uma rede que não seja 802.11, com um esquema de endereçamento ou um formato de quadro diferente, esse serviço cuidará da conversão do formato 802.11 para o formato exigido pela rede de destino.

Os quatro serviços restantes são serviços intracélula (ou intracelulares, isto é, se relacionam a ações dentro de uma única célula). Eles são usados depois que ocorre a associação, e são descritos a seguir.

1. **Autenticação.** Como a comunicação sem fio pode ser enviada ou recebida facilmente por estações não autorizadas, uma estação deve se autenticar antes de ter permissão para transmitir dados. Depois que uma estação móvel é associada pela estação base (ou seja, é aceita em sua célula), a estação base envia um quadro de desafio especial para ver se a estação móvel conhece a chave secreta (senha) que foi atribuída a ela. A estação móvel demonstra seu conhecimento da chave secreta criptografando o quadro de desafio e transmitindo-o de volta à estação base. Se o resultado for correto, a estação móvel será completamente registrada na célula. No padrão inicial, a estação base não tem de provar sua identidade a estação móvel, mas está sendo desenvolvido um trabalho para reparar esse defeito no padrão.

2. **Desautenticação.** Quando uma estação autenticada anteriormente quer deixar a rede, ela é desautenticada. Depois da desautenticação, a estação não pode mais utilizar a rede.

3. **Privacidade.** Para que as informações enviadas por uma LAN sem fio sejam mantidas confidenciais, elas devem ser criptografadas. Esse serviço administra a criptografia e a descriptografia. O algoritmo de criptografia especificado é o RC4, criado por Ronald Rivest, do M.I.T.

4. **Entrega de dados.** Por fim, a transmissão de dados é o objetivo, e assim o 802.11 oferece naturalmente um meio para transmitir e receber dados. Tendo em

vista que 802.11 foi modelado com base no padrão Ethernet e que a transmissão em uma rede Ethernet não oferece a garantia de ser 100% confiável, a transmissão sobre redes 802.11 também não oferece nenhuma garantia de confiabilidade. As camadas mais altas devem lidar com a detecção e a correção de erros.

Uma célula 802.11 tem alguns parâmetros que podem ser inspecionados e, em alguns casos, ajustados. Eles se relacionam à criptografia, aos intervalos de timeout, às taxas de dados, à freqüência de baliza e assim por diante.

As LANs sem fios baseadas no padrão 802.11 estão começando a se desenvolver em edifícios de escritórios, aeroportos, hotéis, restaurantes e campus universitários de todo o mundo. Assim, espera-se um rápido crescimento dessas redes. Para examinar em detalhes o amplo desenvolvimento do padrão 802.11 @@@na CMU, consulte (Hills, 2001).

[T2] 4.5 Redes sem fios de banda larga

Passamos muito tempo cuidando de ambientes internos. Agora, vamos sair e ver se há algo interessante acontecendo lá fora em relação a redes. O fato é que há muita coisa acontecendo, e boa parte de tudo isso está relacionada aos últimos anos. Com a privatização do sistema de telefonia em muitos países, os concorrentes que disputam as empresas de telefonia com freqüência têm permissão para oferecer serviços locais de voz e Internet de alta velocidade. Sem dúvida, há uma grande demanda por esses serviços. O problema é que estender cabos de fibra, coaxiais ou mesmo de par trançado da categoria 5 até milhões de residências e escritórios é algo proibitivamente dispendioso. O que uma empresa concorrente deve fazer?

A resposta é a rede sem fio de banda larga. Erguer uma grande antena em uma colina fora da cidade e instalar antenas orientadas nos telhados dos clientes é

muito mais fácil e econômico que cavar valas e estender cabos. Desse modo, as empresas de telecomunicações concorrentes têm um grande interesse em fornecer um serviço de comunicação sem fio de vários megabits para voz, Internet, filmes por demanda etc. Como vimos na Figura 2.30, o LMDS foi criado para esse fim. Porém, até recentemente, cada concessionária de telecomunicações elaborava seu próprio sistema. Essa falta de padrões significava que não era possível produzir hardware e software em massa, o que mantinha os preços elevados e a aceitação baixa.

Muitas pessoas na indústria perceberam que ter um padrão de banda larga sem fio era o elemento chave que estava faltando, e assim o IEEE teve de formar um comitê composto por pessoas de empresas importantes e do meio acadêmico para elaborar o padrão. O próximo número disponível no espaço de numeração do 802 era **802.16**, e então o padrão recebeu esse número. O trabalho começou em julho de 1999, e o padrão final foi aprovado em abril de 2002. Oficialmente, o padrão é chamado "Air Interface for Fixed Broadband Wireless Access Systems" (interface aérea para sistemas fixos de acesso sem fio de banda larga). No entanto, algumas pessoas preferem chamá-lo **MAN (Metropolitan Area Network — rede metropolitana) sem fio** ou **loop local sem fio**. Vamos considerar todos esses termos intercambiáveis.

Como alguns dos outros padrões 802, o 802.16 foi fortemente influenciado pelo modelo OSI, inclusive nas (sub)camadas, na terminologia, nas primitivas de serviços e em outros itens. Infelizmente, também como o OSI, ele é bastante complicado. Nas próximas seções, apresentaremos uma breve descrição de alguns elementos de destaque do padrão 802.16, mas esse tratamento está longe de ser completo e omite muitos detalhes. Para obter informações adicionais sobre as redes sem fios de banda larga em geral, consulte (Bolcskei *et al.*, 2001; e Webb, 2001). Se desejar informações sobre o padrão 802.16 em particular,

consulte (Eklund *et al.*, 2002).

[T3] 4.5.1 Comparação entre o 802.11 e o 802.16

Neste momento, você pode estar pensando: Por que criar um novo padrão? Por que não usar apenas o 802.11? Existem algumas razões muito boas para não se usar o 802.11, principalmente porque o 802.11 e o 802.16 resolvem problemas diferentes. Antes de entrarmos na tecnologia do 802.16, vale a pena dizer em poucas palavras por que é necessário novo padrão.

Os ambientes em que as redes 802.11 e 802.16 operam são semelhantes em alguns aspectos, principalmente no fato de terem sido projetadas para fornecer comunicações sem fios de alta largura de banda. Entretanto, as redes também diferem por alguns outros detalhes importantes. Para começar, o 802.16 fornece serviço para edifícios, e edifícios não são móveis. Eles não migram de uma célula para outra com freqüência. Grande parte do 802.11 lida com mobilidade, e nada disso é relevante aqui. Além disso, os edifícios podem ter mais de um computador, uma complicação que não ocorre quando a estação final é um notebook. Como os proprietários de edifícios em geral estão dispostos a gastar muito mais dinheiro para desenvolver as comunicações que os proprietários de notebooks, estão disponíveis rádios melhores. Essa diferença significa que o 802.16 pode usar comunicação full-duplex, algo que o 802.11 evita para manter baixo o custo dos rádios.

Tendo em vista que o 802.16 se estende sobre parte de uma cidade, as distâncias envolvidas podem ser de várias quilômetros, o que significa que a potência percebida na estação base pode variar extensamente de estação para estação. Essa variação afeta a relação sinal/ruído que, por sua vez, define vários esquemas de modulação. A comunicação aberta sobre uma cidade também significa que a segurança e a privacidade são essenciais e obrigatórias.

Além disso, cada célula deve ter muito mais usuários que uma célula típica 802.11, e espera-se que esses usuários utilizem maior largura de banda que um usuário típico do 802.11. Afinal, é raro uma empresa chamar 50 funcionários a uma sala com seus laptops para ver se eles conseguem saturar a rede sem fio 802.11 assistindo a 50 filmes diferentes ao mesmo tempo. Por essa razão, é necessário mais espectro do que as bandas ISM podem fornecer, forçando o 802.16 a operar na faixa de freqüências de 10 a 66 GHz, muito mais alta; essa banda é o único lugar do espectro ainda disponível.

Porém, essas ondas milimétricas têm propriedades físicas diferentes das ondas mais longas das bandas ISM o que, por sua vez, exige uma camada física bem diferente. Uma propriedade das ondas milimétricas é o fato de elas serem fortemente absorvidas pela água (em especial pela chuva; mas, até certo ponto, também por neve, granizo e, com um pouco de falta de sorte, pelo nevoeiro intenso). Conseqüentemente, o tratamento de erros é mais importante que em ambientes internos. As ondas milimétricas podem ser concentradas em feixes direcionais (o 802.11 é omnidirecional), e assim escolhas feitas no 802.11 relativas à propagação em vários caminhos são discutíveis aqui.

Outra questão é a qualidade do serviço. Embora o 802.11 forneça algum suporte para tráfego em tempo real (com a utilização do modo PCF), na realidade, ele não foi projetado para telefonia e uso pesado de multimídia. Em contraste, o 802.16 deverá dar suporte completo a essas aplicações, porque foi criado para uso residencial e comercial.

Em resumo, o 802.11 foi projetado para ser a Ethernet móvel, enquanto o 802.16 foi projetado para ser uma rede de televisão a cabo sem fio, mas estacionária. Essas diferenças são tão grandes que os padrões resultantes são muito diferentes, pois eles procuram otimizar aspectos distintos das redes.

Também vale a pena fazermos uma breve comparação com o sistema de telefonia

celular. No caso dos telefones móveis, estamos nos referindo a estações móveis de banda estreita, orientadas para voz e de baixa energia, que se comunicam usando microondas de comprimento médio. Ninguém assiste a filmes de duas horas em alta resolução no seu telefone celular GSM (ainda). Até mesmo o UMTS tem pouca esperança de mudar essa situação. Em resumo, o mundo das MANs sem fios é muito mais exigente que é o mundo da telefonia móvel; portanto, é necessário um sistema completamente diferente. É interessante perguntar se o 802.16 poderia ser usado para dispositivos móveis no futuro. Ele não foi otimizado para esses dispositivos, mas essa possibilidade existe. No momento, ele se destina a redes sem fios fixas.

[T3] 4.5.2 802.16: a pilha de protocolos

A pilha de protocolos do 802.16 é ilustrada na Figura 4.31. A estrutura geral é semelhante à das outras redes 802, mas tem um número maior de subcamadas. A subcamada inferior lida com a transmissão. O rádio tradicional de banda estreita é usado com esquemas de modulação convencionais. Acima da camada de transmissão física encontra-se uma subcamada de convergência para ocultar as diferentes tecnologias da camada de enlace de dados. Na realidade, o 802.11 também tem um recurso semelhante a esse, mas o comitê optou por não formalizá-lo com um nome do tipo OSI.

[arte: imagem original da p. 305]

[Dísticos]

[1] Camadas superiores

[2] Subcamada de convergência de serviços específicos

Parte comum da subcamada MAC

Subcamada de segurança

[3] Camada de enlace de dados

[4] Subcamada dependente do meio físico

[5] Subcamada de convergência e transmissão

QPSK QAM-16 QAM-64

[6] Camada física

[F]Figura 4.31

[FL] A pilha de protocolos do 802.16

Embora não sejam mostrados na figura, já existe um trabalho bem adiantado para acrescentar ao padrão dois novos protocolos da camada física. O padrão 802.16a admitirá o OFDM na faixa de freqüências de 2 a 11 GHz. O padrão 802.16b irá operar na banda ISM de 5 GHz. Ambas as mudanças são tentativas de se aproximar do padrão 802.11.

A camada de enlace de dados consiste em três subcamadas. A inferior lida com privacidade e segurança, que é muito mais crucial para redes públicas externas que para redes privadas internas. Ela cuida da criptografia, da descriptografia e do gerenciamento de chaves.

Em seguida, vem a parte comum da subcamada MAC. É nessa parte que estão localizados os principais protocolos, como o de gerenciamento de canais. De acordo com o modelo, a estação base controla o sistema. Ela pode programar os canais downstream (isto é, da estação base para o assinante) de modo muito eficiente, e também desempenha um papel importante no gerenciamento dos canais upstream (isto é, do assinante para a estação base). Um recurso incomum da subcamada MAC é que, diferente do que ocorre nas outras redes 802, ela é completamente orientada a conexões, a fim de fornecer garantias de qualidade de serviço para a comunicação de telefonia e de multimídia.

A subcamada de convergência de serviços específicos toma o lugar da subcamada de enlace lógico nos outros protocolos 802. Sua função é definir a interface para

a camada de rede. Uma complicação é que o 802.16 foi projetado para se integrar de modo uniforme com os protocolos de datagramas (por exemplo, PPP, IP e Ethernet) e com o ATM. O problema é que os protocolos de pacotes são protocolos sem conexões, enquanto o ATM é orientado a conexões. Isso significa que toda conexão ATM tem de ser mapeada em uma conexão 802.16, o que a princípio é uma questão simples. Porém, sobre qual conexão 802.16 um pacote IP recebido deve ser mapeado? Esse problema foi tratado nessa subcamada.

[T3] 4.5.3 802.16: a camada física

Como mencionamos antes, uma rede sem fio de banda larga necessita de uma grande fração do espectro, e o único lugar em que podemos encontrá-lo é a faixa de 10 a 66 GHz. Essas ondas milimétricas têm uma propriedade interessante que as microondas mais longas não têm: elas trafegam em linha reta, diferente do som, mas muito semelhante à luz. Em conseqüência disso, a estação base pode ter várias antenas, cada uma apontando para um setor diferente do terreno circundante, como mostra a Figura 4.32. Cada setor tem seus próprios usuários e é bastante independente dos setores adjacentes, algo que não é válido no caso do rádio celular, que é omnidirecional.

[arte: imagem original da p. 306]

[Dísticos]

[1] QAM-64 (6 bits/baud)

[2] QAM-16 (4 bits/baud)

[3] QPSK (2 bits/baud)

[F]Figura 4.32

[FL] O ambiente de transmissão do 802.16

Como a intensidade do sinal na banda milimétrica cai nitidamente com a

distância da estação base, a relação sinal/ruído também cai com a distância da estação base. Por essa razão, o 802.16 emprega três esquemas de modulação diferentes, dependendo da distância a que a estação do assinante se encontra em relação à estação base. Para assinantes próximos, é usado o QAM-64, com 6 bits/baud. No caso de assinantes situados a uma distância média, é usado QAM-16, com 4 bits/baud. Para assinantes distantes, é usado o QPSK, com 2 bits/baud. Por exemplo, para um valor típico de 25 MHz do espectro, o QAM-64 oferece 150 Mbps, o QAM-16 oferece 100 Mbps, e o QPSK oferece 50 Mbps. Em outras palavras, quanto mais distante estiver o assinante em relação à estação base, mais baixa será a taxa de dados (semelhante ao que vimos no caso da ADSL na Figura 2.27). Os diagramas de constelação correspondentes a essas três técnicas de modulação foram apresentados na Figura 2.25.

Dado o objetivo de produzir um sistema de banda larga, e considerando as restrições físicas anteriores, os projetistas do 802.16 trabalharam intensamente para usar de forma eficiente o espectro disponível. Ele não gostaram do funcionamento do GSM e do D-AMPS. Ambos utilizam bandas de freqüência distintas, mas equivalentes, para tráfego upstream e downstream. No caso de voz, o tráfego provavelmente é simétrico em sua maior parte; porém, para acesso à Internet, em geral existe maior tráfego downstream do que upstream. Conseqüentemente, o 802.16 fornece um modo mais flexível de alocar a largura de banda. São usados dois esquemas, a **FDD (Frequency Division Duplexing — duplexação por divisão de freqüência)** e **TDD (Time Division Duplexing — duplexação por divisão de tempo)**. Essa última é ilustrada na Figura 4.33. Aqui, a estação base transmite quadros periodicamente. Cada quadro contém slots de tempo. Os primeiros se destinam ao tráfego downstream. Em seguida, há um tempo de proteção usado pelas estações para comutar o sentido. Por fim, temos os slots para tráfego upstream. O número de slots de tempo dedicados a cada

sentido pode ser alterado dinamicamente, a fim de fazer a largura de banda em cada sentido corresponder ao tráfego nesse sentido.

[arte: imagem original da p. 307]

[Dísticos]

[1] Quadro 1 Quadro 2 Quadro 3

[2] Downstream Upstream Tempo de proteção Slot de tempo

[F]Figura 4.33

[FL] Quadros e slots de tempo para duplexação por divisão de tempo

O tráfego downstream é mapeado em slots de tempo pela estação base. A estação base tem o controle completo para esse sentido. O tráfego upstream é mais complexo e depende da qualidade de serviço exigida. Estudaremos a alocação de slots quando descrevermos a subcamada MAC a seguir.

Outra característica interessante da camada física é sua habilidade para reunir vários quadros MAC enfileirados em uma única transmissão física. Esse recurso aumenta a eficiência espectral, reduzindo o número de preâmbulos e cabeçalhos da camada físicos necessários.

Também vale a pena notar o uso de códigos de Hamming para efetuar a correção antecipada de erros na camada física. Quase todas as outras redes simplesmente empregam totais de verificação para detectar erros e solicitam a retransmissão quando os quadros são recebidos com erros. Porém, no ambiente de banda larga geograficamente distribuído, esperam-se tantos erros de transmissão que a correção de erros é empregada na camada física, além dos totais de verificação das camadas mais altas. O efeito final da correção de erros é fazer o canal parecer melhor do que realmente é (da mesma forma que os discos de CD-ROM parecem ser muito confiáveis, mas apenas porque mais da metade da quantidade total de bits é dedicada à correção de erros na camada física).

[T3] 4.5.4 802.16: o protocolo da subcamada MAC

A camada de enlace de dados é dividida em três subcamadas, como vimos na Figura 4.31. Tendo em vista que só estudaremos a criptografia no Capítulo 8, é difícil explicar agora como funciona a subcamada de segurança. Basta saber que a criptografia é usada para manter secretos todos os dados transmitidos. Apenas a carga útil de cada quadro é criptografada; os cabeçalhos não são. Essa propriedade significa que um espião pode ver quem está se comunicando com quem, mas não consegue saber o que uma pessoa está dizendo à outra.

Se você já conhece algo sobre criptografia, aqui está uma explicação de apenas um parágrafo sobre a subcamada de segurança. Se não souber nada sobre criptografia, é provável que você não considere o próximo parágrafo muito esclarecedor (mas talvez fosse interessante ler outra vez esse parágrafo depois de concluir o Capítulo 8).

No momento em que um assinante se conecta a uma estação base, eles executam um processo de autenticação mútua com criptografia RSA de chave pública, usando certificados X.509. As cargas úteis propriamente ditas são criptografadas com a utilização de um sistema de chave simétrica, seja ele o DES com encadeamento de blocos de cifras ou o DES triplo com duas chaves. O AES (Rijndael) deverá ser acrescentado em breve. A verificação de integridade emprega o SHA-1. Não foi tão ruim assim, certo?

Agora, vamos examinar a parte comum da subcamada MAC. Os quadros MAC ocupam um número inteiro de slots de tempo da camada física. Cada quadro é composto por subquadros, sendo os dois primeiros os mapas downstream e upstream. Esses mapas informam o que existe em cada slot de tempo e quais slots de tempo estão livres. O mapa downstream também contém vários parâmetros do sistema, a fim de informá-los às novas estações quando elas se

conectarem.

O canal downstream é bastante direto. A estação base simplesmente decide o que inserir em cada subquadro. O canal upstream é mais complicado, pois existem assinantes concorrentes não coordenados que precisam de acesso a ele. Sua alocação está intimamente relacionada à questão da qualidade de serviço. São definidas quatro classes de serviço, da seguinte forma:

1. Serviço de taxa de bits constante.
2. Serviço de taxa de bits variável de tempo real.
3. Serviço de taxa de bits variável não de tempo real.
4. Serviço de @@@melhor esforço.

Todo serviço no 802.16 é orientado a conexões, e cada conexão recebe uma das classes de serviço anteriores, determinada quando a conexão é configurada. Essa estrutura é muito diferente da estrutura do 802.11 ou da Ethernet, que não têm conexões na subcamada MAC.

O serviço de taxa de bits constante se destina à transmissão de voz não compactada, como em um canal T1. Esse serviço precisa enviar uma quantidade de dados predeterminada a intervalos de tempo predeterminados. Ele é acomodado dedicando-se certos slots de tempo a cada conexão desse tipo. Uma vez que a largura de banda é alocada, os slots de tempo ficam disponíveis automaticamente, sem a necessidade de solicitar cada um.

O serviço de taxa de bits variável de tempo real se destina a aplicações de multimídia compactada e a outras aplicações de software de tempo real em que a quantidade de largura de banda necessária em cada instante pode variar. Ele é acomodado fazendo-se a estação base consultar o assinante a intervalos fixos sobre a quantidade de largura de banda necessária em cada momento.

O serviço de taxa de bits variável não de real tempo se destina a transmissões pesadas que não são de tempo real, como as transferências de grandes arquivos.

Para esse serviço, a estação base consulta o assinante com freqüência, mas não efetua o polling a intervalos de tempo prescritos com rigidez. Um cliente de taxa de bits constante pode definir um bit em um de seus quadros solicitando uma consulta para transmitir tráfego adicional (de taxa de bits variável).

Se uma estação não responder a uma consulta *k* vezes seguidas, a estação base a colocará em um grupo de multidifusão, retirando-a de seu polling pessoal. Em vez disso, quando o grupo de multidifusão for consultado, qualquer das estações que ele contém poderá responder, disputando o serviço. Desse modo, estações com tráfego não irão desperdiçar valiosos períodos de polling.

Por fim, o serviço de @@@melhor esforço se destina a todos os outros casos. Nenhum polling é feito e o assinante deve disputar a largura de banda com outros assinantes do serviço de @@@melhor esforço. As solicitações de largura de banda são feitas em slots de tempo marcados no mapa upstream como disponíveis para disputa. Se uma solicitação for bem sucedida, seu sucesso será notado no próximo mapa downstream. Se ela tiver sucesso, os assinantes malsucedidos terão de tentar de novo mais tarde. Para minimizar colisões, é usado o algoritmo de recuo binário exponencial da Ethernet.

O padrão define duas formas de alocação de largura de banda: por estação e por conexão. No primeiro caso, a estação do assinante agrega as necessidades de todos os usuários no edifício e faz solicitações coletivas para eles. Quando a largura de banda é concedida, a estação reparte essa largura de banda entre seus usuários, conforme seus critérios. No último caso, a estação base administra diretamente cada conexão.

[T3] 4.5.5 802.16: estrutura de quadro

Todos os quadros MAC começam com um cabeçalho genérico. O cabeçalho é seguido por um carga útil opcional e um total de verificação (CRC) opcional, como

ilustra a Figura 4.34. A carga útil não é necessária em quadros de controle como, por exemplo, aqueles que solicitam slots de canais. O total de verificação (de forma surpreende) também é opcional, devido à correção de erros na camada física e ao fato de não ser feita nenhuma tentativa de retransmitir quadros de tempo real. Por que se preocupar com um total de verificação se não haverá nenhuma tentativa de retransmissão?

[arte: imagem original da p. 309]

[Dísticos]

[1] Bits 1 1 6 1 1 2 1 11 16 8 4

[2] (a) 0 EC Tipo CI EK Comprimento ID de conexão CRC de cabeçalho Dados CRC

[3] Bits 1 1 6 16 16 8

[4] (b) 1 0 Tipo Bytes necessários ID de conexão CRC de cabeçalho

[F]Figura 4.34

[FL] (a) Um quadro genérico. (b) Um quadro de solicitação de largura de banda

Apresentaremos a seguir um breve resumo de informações sobre os campos do cabeçalho da Figura 4.34(a). O bit *EC* informa se a carga útil está criptografada. O campo *Tipo* identifica o tipo de quadro, informando principalmente se a compactação e a fragmentação estão presentes. O campo *CI* indica a presença ou a ausência do total de verificação final. O campo *EK* informa qual das chaves de criptografia está sendo usada (se houver). O campo *Comprimento* fornece o comprimento completo do quadro, incluindo o cabeçalho. O *Identificador de conexão* informa a qual conexão esse quadro pertence. Por fim, o campo *CRC de cabeçalho* é um total de verificação relativo apenas ao cabeçalho, empregando o polinômio $x^8 + x^2 + x + 1$.

Um segundo tipo de cabeçalho, para quadros que solicitam largura de banda, é

mostrado na Figura 4.34(b). Ele começa com um bit 1 em vez de um bit 0 e é semelhante ao cabeçalho genérico, exceto pelo fato de que o segundo e o terceiro bytes formam um número de 16 bits que informa a quantidade de largura de banda necessária para transportar o número especificado de bytes. Os quadros de solicitação de largura de banda não transportam uma carga útil ou um CRC para um quadro inteiro.

Poderia ser dito muito mais sobre o padrão 802.16, mas este não é o lugar apropriado. Para obter mais informações, consulte o próprio padrão.

[T2] 4.6 Bluetooth

Em 1994, a empresa L. M. Ericsson ficou interessada em conectar seu telefones móveis a outros dispositivos (por exemplo, PDAs) sem cabos. Junto com outras quatro empresas (IBM, Intel, Nokia e Toshiba), ela formou um SIG (Special Interest Group, isto é, consórcio) com o objetivo de desenvolver um padrão sem fio para interconectar dispositivos de computação e comunicação e ainda acessórios, utilizando rádios sem fios de curto alcance, baixa potência e baixo custo. O projeto foi denominado **Bluetooth** em homenagem a Harald Blaatand (Bluetooth) II (940-981), um rei viking que unificou (isto é, conquistou) a Dinamarca e a Noruega, também sem cabos.

Embora a idéia original fosse apenas se livrar dos cabos entre dispositivos, ela logo começou a expandir seu escopo e invadir a área das LANs sem fios. Embora essa mudança torne o padrão mais útil, também cria alguma competição pelo mercado com o 802.11. Para piorar, os dois sistemas também interferem eletricamente um com o outro. Também vale a pena notar que a Hewlett-Packard introduziu uma rede de infravermelho para conectar periféricos de computadores sem fios há alguns anos, mas ele nunca obteve realmente um grande êxito.

Sem temer tudo isso, em julho de 1999, o consórcio do Bluetooth emitiu uma

especificação de 1500 páginas da versão 1.0. Pouco empo depois, o grupo de padrões no IEEE que examinava as redes especiais sem fios, o 802.15, adotou o documento do Bluetooth como base e começou a modificá-lo. Embora pareça estranho algo que já tinha uma especificação muito detalhada e nenhuma implementação incompatível que precisasse ser harmonizada, a história mostra que a existência de um padrão aberto gerenciado por um corpo neutro como o IEEE com freqüência promove o uso de uma tecnologia. Para ser um pouco mais preciso, devemos observar que a especificação do Bluetooth se refere a um sistema completo, desde a camada física até a camada de aplicação. O comitê 802.15 do IEEE está padronizando apenas as camadas física e de enlace de dados; o restante da pilha de protocolos foi ignorada.

Embora o IEEE tenha aprovado em 2002 o primeiro padrão PAN, o 802.15.1, o SIG do Bluetooth ainda está ativo, ocupado com melhorias. Embora as versões de Bluetooth do SIG e do IEEE não sejam idênticas, espera-se que elas logo venham a convergir para um único padrão.

[T3] 4.6.1 Arquitetura do Bluetooth

Vamos começar nosso estudo do sistema Bluetooth com uma avaliação rápida do que ele contém e do que planeja fazer. A unidade básica de um sistema Bluetooth é uma **piconet**, que consiste em um nó mestre e até sete nós escravos ativos, situados dentro de uma distância de 10 metros. Podem existir muitas piconets na mesma sala (grande) e elas podem até mesmo ser conectadas por um nó de ponte, como mostra a Figura 4.35. Uma coleção interconectada de piconets é chamada **scatternet.**

[arte: ver original p. 311]

[Dísticos]

[1] Piconet 1

[2] Escravo ativo

[3] Piconet 2

[4] Escravo inativo (estacionado)

[5] Escravo ponte

[F]Figura 4.35

[FL] Duas piconets podem ser conectadas para formar uma scatternet

Além dos sete nós escravos ativos em uma piconet, pode haver até 255 nós estacionados (inativos) na rede. Esses nós são dispositivos que o mestre comutou para um estado de baixa energia, a fim de reduzir o consumo em suas baterias. No estado estacionado, um dispositivo não pode fazer nada, exceto responder a um sinal de ativação ou de baliza do mestre. Também existem dois estados de energia intermediários, @@@hold e sniff, mas esses estados não serão estudados aqui.

A razão para a estrutura de mestre/escravo é que os projetistas pretendiam facilitar a implementação de chips Bluetooth completos por menos de 5 dólares. Em conseqüência dessa decisão, os escravos são "não inteligentes", fazendo basicamente apenas o que o mestre determina. Em seu núcleo, uma piconet é um sistema TDM centralizado, no qual o mestre controla o clock e define qual dispositivo irá se comunicar em cada slot de tempo. Toda comunicação é feita entre o mestre e um escravo; não é possível a comunicação direta entre escravos.

[T3] 4.6.2 Aplicações do Bluetooth

A maioria dos protocolos de rede só fornece canais entre entidades que se comunicam, deixando para os projetistas de aplicações a tarefa de descobrir a utilidade desses canais. Por exemplo, o 802.11 não especifica se os usuários devem usar seus notebooks para ler correio eletrônico, navegar na Web ou

qualquer outra ação. Em contraste, a especificação Bluetooth V1.1 identifica 13 aplicações específicas que serão admitidas e fornece diferentes pilhas de protocolos para cada uma. Infelizmente, essa abordagem aumentou muito a complexidade, o que omitiremos aqui. As 13 aplicações, chamadas **perfis**, estão listadas na Figura 4.36. Examinando-as rapidamente agora, podemos ver de modo mais claro o que o SIG do Bluetooth está tentando realizar.

[arte: ver original p. 312]

[T]Tabela

Nome	**Descrição**
Acesso genérico	Procedimentos para gerenciamento de enlaces
Descoberta de serviço	Protocolo para descobrir serviços oferecidos
Porta serial	Substitui um cabo de porta serial
Intercâmbio genérico de objetos	Define o relacionamento cliente/servidor para movimentação de objetos
Acesso de LAN	Protocolo entre um computador móvel e uma LAN fixa
Rede dial-up	Permite que um notebook se conecte através de um telefone móvel
Fax	Permite que um equipamento de fax móvel se comunique com um telefone móvel
Telefonia sem fio	Conecta um aparelho telefônico à sua estação base local
Intercomunicador	Intercomunicação digital
Fone de ouvido	Permite a comunicação de voz sem o uso da mãos
Push de objetos	Fornece um meio para intercambiar objetos simples
Transferência de arquivos	Fornece um recurso mais geral de transferência de arquivos
Sincronização	Permite sincronizar um PDA com outro computador

[F]Figura 4.36

[FL] Os perfis do Bluetooth

O perfil de acesso genérico não é realmente uma aplicação, mas sim a base sobre a qual são elaboradas as aplicações reais. Sua principal função é fornecer um meio para estabelecer e manter enlaces seguros (canais) entre o mestre e os escravos. Também é relativamente genérico o perfil de descoberta de serviço, utilizado pelos dispositivos para descobrir quais são os serviços que outros dispositivos têm a oferecer. Espera-se que todos os dispositivos do Bluetooth implementem esses dois perfis. Os restantes são opcionais.

O perfil de porta serial é um protocolo de transporte utilizado pela maioria dos outros perfis. Ele emula uma linha serial e é especialmente útil para aplicações de tecnologia antiga que esperam encontrar uma linha serial.

O perfil de intercâmbio genérico de objetos define um relacionamento cliente/servidor para movimentação de dados. Os clientes iniciam operações, mas um escravo pode ser um cliente ou um servidor. Como o perfil de porta serial, esse é um bloco de construção que serve de base para outros perfis.

O grupo seguinte tem três perfis relacionados às redes. O perfil de acesso de LAN permite que um dispositivo Bluetooth se conecte a uma rede fixa. Esse perfil é um concorrente direto do 802.11. O perfil de rede dial-up foi a motivação original de todo o projeto. Ele permite que um notebook se conecte a um telefone móvel contendo um modem interno sem fios. O perfil de fax é semelhante ao de rede dial-up, exceto por permitir que equipamentos de fax sem fios transmitam e recebam mensagens de fax usando telefones móveis, sem um fio interligando os dois aparelhos.

Os três perfis seguintes se referem à telefonia. O perfil de telefonia sem fio fornece um meio para conectar o aparelho de um telefone sem fio à estação base. Atualmente, a maioria dos telefones sem fios não pode ser usada também como

telefone móvel mas, no futuro, os telefones sem fios e os telefones móveis deverão se fundir. O perfil de intercomunicação permite que dois telefones se conectem como intercomunicadores. Por fim, o perfil de fone de ouvido proporciona comunicação de voz sem o uso das mãos entre o fone de ouvido e sua estação base para, por exemplo, tornar possível a comunicação telefônica enquanto se dirige um automóvel.

Na realidade, os três perfis restantes se destinam à troca de objetos entre dois dispositivos sem fios. Esses objetos poderiam ser cartões de visita, figuras ou arquivos de dados. Em particular, o perfil de sincronização será usado com a finalidade de transferir dados para um PDA ou notebook quando o usuário sair de casa, e para coletar dados desses equipamentos quando o usuário retornar.

Seria realmente necessário explicar todas essas aplicações em detalhes e fornece diferentes pilhas de protocolos para cada uma? É provável que não, mas surgiram diversos grupos de trabalho que elaboraram partes distintas do padrão, e cada um se concentrou em seu problema específico e gerou seu próprio perfil. Imagine tudo isso como uma aplicação da lei de Conway. (Na edição de abril de 1968 da revista *Datamation*, Melvin Conway observou que, se designar *n* pessoas para escrever um compilador, você obterá um compilador de *n* passagens ou, de modo mais geral, a estrutura de software reflete a estrutura do grupo que o produziu.) Provavelmente, teria sido possível concluir o trabalho com duas pilhas de protocolos em vez de 13, uma para transferência de arquivos e uma para comunicação de fluxo em tempo real.

[T3] 4.6.3 A pilha de protocolos do Bluetooth

O padrão Bluetooth tem muitos protocolos agrupados livremente em camadas. A estrutura de camadas não segue o modelo OSI, o modelo TCP/IP, o modelo 802 ou qualquer outro modelo conhecido. Porém, o IEEE está trabalhando na

modificação do Bluetooth para adaptá-lo melhor ao modelo 802. A arquitetura básica de protocolos do Bluetooth, modificada pelo comitê 802, está representada na Figura 4.37.

A camada inferior é a camada física de rádio, que corresponde muito bem à camada física nos modelos OSI e 802. Ela lida com a transmissão e a modulação de rádio. Muitas das preocupações aqui estão relacionadas ao objetivo de tornar o sistema mais econômico, para que possa vir a ser um item do mercado de massa.

A camada de banda base é de certa forma análoga à subcamada MAC, mas também inclui elementos da camada física. Ela lida com a maneira como o mestre controla os slots de tempo e como esses slots são agrupados em quadros.

[arte: ver original p. 314]

[Dísticos]

[1] Aplicações/Perfis

[2] Áudio Outras RFcomm Telefonia Descoberta de serviços Controle

LLC

Protocolo de adaptação de controle de enlace lógico

Gerenciador de enlaces

[3] Banda base

Camada física de rádio

[4] Camada de aplicação

Camada de middleware

Camada de enlace de dados

Camada física

[F]Figura 4.37

[FL] A versão 802.15 da arquitetura de protocolos do Bluetooth

Em seguida, temos uma camada com um grupo de protocolos até certo ponto inter-relacionados. O gerenciador de enlaces cuida do estabelecimento de canais lógicos entre dispositivos, incluindo o gerenciamento de energia, autenticação e qualidade de serviço. O protocolo de adaptação de controle de enlace lógico (freqüentemente chamado L2CAP) isola as camadas superiores dos detalhes de transmissão. Ele é análogo à subcamada LLC do padrão 802, mas é tecnicamente diferente dela. Como seus nomes sugerem, os protocolos de áudio e controle lidam respectivamente com o áudio e o controle. As aplicações podem chegar até eles diretamente, sem terem de passar pelo protocolo L2CAP.

A próxima camada é a camada middleware, que contém uma mistura de diferentes protocolos. O LLC do 802 foi inserido aqui pelo IEEE para manter a compatibilidade com as outras redes 802. Os protocolos RFcomm, de telefonia e de descobertas de serviços são originais. O protocolo RFcomm (comunicação por freqüência de rádio) é o protocolo que emula a porta serial padrão encontrada nos computadores pessoais para conectar o teclado, o mouse e o modem, entre outros dispositivos. Ele foi projetado para permitir que dispositivos de tecnologia antiga o utilizem com facilidade. O protocolo de telefonia é um protocolo de tempo real utilizado pelos três perfis orientados para voz. Ele também gerencia a configuração e o encerramento de chamadas. Por fim, o protocolo de descoberta de serviços é usado para localizar serviços na rede.

As aplicações e os perfis se localizam na camada superior. Eles utilizam os protocolos das camadas inferiores para cumprir suas funções. Cada aplicação tem seu próprio subconjunto dedicado dos protocolos. Dispositivos específicos, como um fone de ouvido, em geral só contêm os protocolos exigidos por essa aplicação e nenhum outro.

Nas próximas seções, examinaremos as três camadas mais baixas da pilha de

protocolos do Bluetooth, pois elas correspondem aproximadamente à camada física e à subcamada MAC .

[T4] 4.6.4 A camada de rádio do Bluetooth

A camada de rádio move os bits do mestre para o escravo ou vice-versa. Ela é um sistema de baixa potência com um alcance de 10 metros, operando na banda ISM de 2,4 GHz. A banda está dividida em 79 canais de 1 MHz cada uma. A modulação é de chaveamento por deslocamento de freqüência, com 1 bit por Hz, fornecendo uma taxa de dados bruta igual a 1 Mbps, mas grande parte desse espectro é consumido por overhead. Para alocar os canais de maneira uniforme, é usado o espectro de dispersão de saltos de freqüência com 1600 hops/s e um tempo de parada de 625 µs. Todos os nós em uma piconet saltam simultaneamente, com o mestre ditando a seqüência de saltos.

Como o 802.11 e o Bluetooth operam na banda ISM de 2,4 GHz nos mesmos 79 canais, eles interferem um com o outro. Tendo em vista que o Bluetooth salta com muito maior rapidez que o 802.11, é muito mais provável que um dispositivo Bluetooth arruine as transmissões do 802.11 que o contrário. Como o 802.11 e o 802.15 são ambos padrões IEEE, o IEEE está procurando uma solução para esse problema, mas não é tão fácil encontrá-la, pois ambos os sistemas utilizam a banda ISM pela mesma razão: não é exigido nenhum licenciamento nessa banda. O padrão 802.11a usa a outra banda ISM (5 GHz), mas tem um alcance muito mais curto que o 802.11b (devido a aspectos físicos das ondas de rádio), e assim a utilização do 802.11a não é uma solução perfeita em todos os casos. Algumas empresas resolveram o problema proibindo completamente o Bluetooth. Uma solução baseada no mercado é adotar a rede com maior poder (político e econômico, não elétrico), a fim de obrigar a parte mais fraca a modificar seu padrão para parar de interferir com o outro. Algumas reflexões

sobre esse tema encontram-se em (Lansford *et al.*, 2001).

[T3] 4.6.5 A camada de banda base do Bluetooth

A camada de banda base é a estrutura mais próxima de uma subcamada MAC que o Bluetooth tem. Ela transforma o fluxo bruto de bits em quadros e define alguns formatos importantes. Em sua forma mais simples, o mestre de cada piconet define uma série de slots de tempo de 625 μs, com as transmissões do mestre começando nos slots pares e as transmissões dos escravos começando nos slots ímpares. Essa é tradicional multiplexação por divisão de tempo, em que o mestre fica com metade dos slots e os escravos compartilham a outra metade. Os quadros podem ter 1, 3 ou 5 slots de duração.

A sincronização por saltos de freqüência permite um tempo de ajuste de 250 a 260 μs por salto, para permitir que os circuitos de rádio se estabilizem. É possível um ajuste mais rápido, mas apenas a um custo mais alto. No caso de um quadro de um único slot, após o ajuste, restam 366 dos 625 bits. Destes, 126 se destinam a um código de acesso e ao cabeçalho, restando 240 bits para dados. Quando cinco slots são reunidos, só é utilizado um período de ajuste um pouco mais curto e, assim, dos $5 \times 625 = 3125$ bits em cinco slots de tempo, 2781 ficam disponíveis para a camada de banda base. Desse modo, os quadros mais longos são muito mais eficientes que quadros de um único slot.

Cada quadro é transmitido sobre um canal lógico, chamado **enlace (link)**, entre o mestre e um escravo. Há dois tipos de enlaces. O primeiro é o enlace **ACL (Asynchronous Connection-Less — assíncrono sem conexões)**, usado para dados comutados por pacotes disponíveis a intervalos irregulares. Esses dados vêm da camada L2CAP no lado de transmissão e são entregues à camada L2CAP no lado de recepção. O tráfego ACL é entregue em uma base de @@@melhor esforço. Não é dada nenhuma garantia. Os quadros podem ser perdidos e pode ser necessário

retransmiti-los. Um escravo só pode ter um enlace ACL para seu mestre.

O outro é o enlace **SCO (Synchronous Connection Oriented — síncrono orientado a conexões)**, para dados de tempo real, como as conexões telefônicas. A esse tipo de canal é alocado um slot fixo em cada sentido. Devido à natureza crítica dos enlaces SCO, os quadros enviados sobre eles nunca são retransmitidos. Em vez disso, pode ser usada a correção de erros antecipada para proporcionar alta confiabilidade. Um escravo pode ter até três enlaces SCO com seu mestre. Cada enlace SCO pode transmitir um canal de áudio PCM de 64.000 bps.

[T3] 4.6.6 A camada L2CAP do Bluetooth

A camada L2CAP tem três funções importantes. Primeiro, ela aceita pacotes de até 64 KB das camadas superiores e os divide em quadros para transmissão. Na outra extremidade, os quadros são montados novamente em pacotes.

Em segundo lugar, ela lida com a multiplexação e a demultiplexação de várias origens de pacotes. Quando um pacote é novamente montado, a L2CAP determina a qual protocolo da camada superior ele será entregue; por exemplo, RFcomm ou telefonia.

Em terceiro lugar, a camada L2CAP lida com os requisitos de qualidade de serviço, tanto quando os enlaces são estabelecidos quanto durante a operação normal. Também é negociado em tempo de configuração o tamanho máximo de carga útil permitido, a fim de impedir que um dispositivo de pacotes grandes afogue um dispositivo de pacotes pequenos. Esse recurso é necessário, porque nem todos os dispositivos podem manipular o pacote máximo de 64 KB.

[T3] 4.6.7 A estrutura de quadro do Bluetooth

Há vários formatos de quadros; o mais importante é apresentado na Figura 4.38. Ele começa com um código de acesso que normalmente identifica o mestre, para

que os escravos situados dentro do alcance de rádio de dois mestres possam conhecer o destino de cada tráfego. Em seguida, há um cabeçalho de 54 bits contendo campos típicos da subcamada MAC. Depois, vem o campo de dados, de até 2744 bits (no caso de uma transmissão de cinco slots). Para um único slot de tempo, o formato é o mesmo, exceto pelo fato do campo de dados ter 240 bits.

[arte: ver original p. 316]

[Dísticos]

[1] Bits 72 54 0 a 2744

[2] Código de acesso Cabeçalho Dados

[3] Bits 3 4 1 1 1 8

[4] Ender. Tipo F A S Total de verificação

[5] O cabeçalho de 18 bits é repetido três vezes, dando um total de 54 bits

[F]Figura 4.38

[FL] Um quadro de dados típico do Bluetooth

Vamos examinar rapidamente o cabeçalho. O campo *Endereço* identifica qual dos oito dispositivos ativos é o destino do quadro. O campo *Tipo* identifica o tipo de quadro (ACL, SCO, polling ou nulo), o tipo de correção de erros usado no campo de dados, e de quantos slots é a duração do quadro. O bit *Fluxo* é definido por um escravo quando seu buffer está cheio e não pode receber mais dados. Essa é uma forma primitiva de controle de fluxo. O bit *Confirmação* é usado para transportar uma mensagem ACK em um quadro. O bit *Seqüência* é usado para numerar os quadros, a fim de detectar retransmissões. O protocolo é stop-and-wait, e assim 1 bit é suficiente. Em seguida, temos o cabeçalho de 8 bits *Total de verificação*. O cabeçalho de 18 bits inteiro é repetido três vezes para formar o cabeçalho de 54 bits mostrado na Figura 4.38. No lado receptor, um circuito simples examina todas as três cópias de cada bit. Se todas três forem iguais, o bit

será aceito. Se não, vence a opinião da maioria. Desse modo, 54 bits de capacidade de transmissão são usados para enviar 10 bits de cabeçalho. A razão para isso é que, para transmitir dados de maneira confiável em um ambiente ruidoso usando dispositivos de baixo custo e de baixa potência (2,5 mW) com pouca capacidade de computação, é necessária uma grande redundância.

São usados vários formatos para o campo de dados de quadros ACL. Entretanto, os quadros SCO são mais simples: o campo de dados tem sempre 240 bits. São definidas três variantes, permitindo 80, 160 ou 240 bits de carga útil real, sendo os bits restantes usados para correção de erros. Na versão mais confiável (carga útil de 80 bits), o conteúdo é simplesmente repetido três vezes, da mesma forma que o cabeçalho.

Tendo em vista que o escravo só pode usar os slots ímpares, ele recebe consegue 800 slots/s, da mesma maneira que o mestre. Com uma carga útil de 80 bits, a capacidade de canal do escravo é de 64.000 bps, e a capacidade de canal do mestre também é de 64.000 bps, exatamente o bastante para um único canal de voz PCM full-duplex (e esse é o motivo de ter sido escolhida uma taxa de saltos de 1600 hops/s). Esses números significam que um canal de voz full-duplex com 64.000 bps em cada sentido usando o formato mais confiável satura completamente a piconet, apesar de uma largura de banda bruta de 1 Mbps. No caso da variante menos confiável (240 bits/slot sem redundância nesse nível), podem ser admitidos três canais de voz full-duplex ao mesmo tempo, e esse é o motivo de ser permitido um máximo de três enlaces SCO por escravo.

Existe muito mais a ser dito sobre o Bluetooth, mas não há mais espaço para isso aqui. Se desejar mais informações, consulte (Bhagwat, 2001; Bisdikian, 2001; Bray e Sturman, 2002; Haartsen, 2000; Johansson *et al.*, 2001; Miller e Bisdikian, 2001; e Sairam *et al.*, 2002).

[T2] 4.7 Comutação na camada de enlace de dados

Muitas empresas têm diversas LANs e desejam conectá-las. As LANs podem ser conectadas por dispositivos chamados **pontes**, que operam na camada de enlace de dados. As pontes examinam os endereços da camada de enlace de dados para efetuar o roteamento. Tendo em vista que elas não têm de examinar o campo de carga útil dos quadros que roteiam, as pontes podem transportar o IPv4 (usado na Internet hoje), o IPv6 (que será usado na Internet do futuro), o AppleTalk, o ATM, o OSI ou quaisquer outros tipos de pacotes. Em contraste, os *roteadores* examinam os endereços em pacotes e efetuam o roteamento com base nesses endereços. Embora pareça haver uma clara distinção entre pontes e roteadores, alguns desenvolvimentos modernos, como o advento da Ethernet comutada, confundiram essas definições, como veremos mais adiante. Nas próximas seções, examinaremos as pontes e os switches, especialmente para conectar LANs 802 diferentes. Para conhecer um tratamento completo de pontes, switches e tópicos relacionados, consulte (Perlman, 2000).

Antes de iniciarmos o estudo da tecnologia de pontes, vale a pena examinarmos algumas situações comuns em que elas são usadas. Mencionaremos seis razões pelas quais uma única organização pode ter várias LANs.

Primeiro, muitas universidades e departamentos de empresas têm suas próprias LANs, principalmente para conectar seus computadores pessoais, estações de trabalho e servidores. Como os objetivos dos departamentos são diferentes, muitos deles escolhem LANs distintas, sem se importar com o que outros departamentos estão fazendo. Mais cedo ou mais tarde, surge a necessidade de interação; por isso as pontes são necessárias. Neste exemplo, a existência de diversas LANs se deve à autonomia de seus proprietários.

Em segundo lugar, a organização pode estar geograficamente dispersa em vários edifícios separados por distâncias consideráveis. Talvez seja mais econômico ter

LANs separadas em cada edifício e conectá-las com pontes e enlaces de laser que estender um único cabo sobre toda a instalação.

[arte: imagem original da p. 318]

[Dísticos]

[1] LAN de backbone

B B

Servidor de arquivos

Estação de trabalho

[2] Ponte

B B

Agrupamento em uma única LAN

LAN

[F]Figura 4.39

[FL] Diversas LANs conectadas por um backbone para tratar uma carga total maior do que a capacidade de uma única LAN

Em terceiro lugar, talvez seja necessário dividir aquilo que é logicamente uma única LAN em LANs separadas, a fim de acomodar a carga. Por exemplo, em muitas universidades, há milhares de estações de trabalho disponíveis para as necessidades de computação dos funcionários e dos alunos. Os arquivos normalmente são mantidos em máquinas servidoras de arquivos e são transferidos por download para as máquinas dos usuários, se estes assim o solicitarem. A enorme escala desse sistema impossibilita a colocação de todas as estações de trabalho em uma única LAN — a largura de banda total necessária seria excessivamente alta. Em vez disso, são usadas diversas LANs conectadas por pontes, como mostra a Figura 4.39. Cada LAN contém um grupo de estações de trabalho com seu próprio servidor de arquivos, de forma que a maior parte do

tráfego se restringe a uma única LAN, o que não aumenta a carga do backbone. Vale a pena notar que, embora as LANs sejam representadas como cabos multiponto na Figura 4.39 (a representação clássica), elas costumam ser implementadas com maior freqüência com hubs ou switches especiais. Entretanto, um longo cabo multiponto com várias máquinas conectadas a ele e um hub com as máquinas conectadas no interior do hub têm funcionalidade idêntica. Em ambos os casos, todas as máquinas pertencem ao mesmo domínio de colisão, e todas utilizam o protocolo CSMA/CD para transmitir quadros. Porém, as LANs comutadas são diferentes, conforme vimos antes e como veremos de novo em breve.

Em quarto lugar, em algumas situações, uma única LAN poderia ser adequada em termos de carga, mas a distância física entre as máquinas mais distantes seria muito grande (por exemplo, mais de 2,5 km para o padrão Ethernet). Mesmo que fosse fácil estender o cabo a rede não funcionaria, devido ao retardo de ida e volta excessivamente longo. A única solução é particionar a LAN e instalar pontes entre os segmentos. Usando pontes, a distância física total coberta pode ser aumentada.

O quinto motivo é a confiabilidade. Em uma única LAN, um nó defeituoso que continua a transmitir um fluxo contínuo de lixo pode danificar a LAN. As pontes podem ser inseridas em trechos críticos, como as portas corta fogo de um edifício, a fim de evitar que um único nó desativado derrube todo o sistema. Ao contrário de um repetidor, que apenas copia o que vê, uma ponte pode ser programada para exercer algum critério sobre o que deve encaminhar e o que não deve encaminhar.

Em sexto e último lugar, as pontes podem contribuir para a segurança da organização. A maioria das interfaces de LANs tem um **modo promíscuo**, no qual *todos* os quadros são enviados ao computador, e não apenas os quadros

endereçados a ele. Os espiões e os intrometidos adoram esse recurso. Com a inserção de pontes em diversos lugares e tendo cuidado para não encaminhar tráfego de natureza delicada, um administrador de sistema pode isolar partes da rede, de forma que seu tráfego não possa escapar e cair em mãos erradas.

No caso ideal, as pontes devem ser totalmente transparentes, significando que deve ser possível mover uma máquina de um segmento de cabo para outro sem alterar qualquer hardware, software ou tabela de configuração. Além disso, deve ser possível a comunicação entre máquinas de qualquer segmento e máquinas de qualquer outro segmento, independente dos tipos de LANs que estejam sendo usadas nos dois segmentos ou em outros segmentos situados entre eles. Às vezes, esse objetivo é alcançado, mas nem sempre.

[T3] 4.7.1 Pontes entre LANs 802.x e 802.y

Depois de verificarmos por que as pontes são necessárias, agora vamos descrever como elas funcionam. A Figura 4.40 ilustra a operação de uma ponte simples de duas portas. O host *A* de uma LAN sem fio (802.11) tem um pacote a ser enviado a um host fixo *B* situado em uma rede Ethernet (802.3), à qual a LAN sem fio está conectada. O pacote desce até a subcamada LLC e adquire um cabeçalho LLC (mostrado em preto na figura). Em seguida, ele passa para a subcamada MAC e recebe um cabeçalho 802.11 (além de um final, não mostrado na figura). Essa unidade continua a se propagar pelo ar e é captada pela estação base, que percebe que ela precisa ir para a Ethernet fixa. Quando o pacote chega à ponte que conecta a rede 802.11 à rede 802.3, ele começa na camada física e segue seu caminho ascendente. Na subcamada MAC da ponte, o cabeçalho 802.11 é retirado. Em seguida, o pacote puro (com o cabeçalho LLC) é levado até a subcamada LLC da ponte. Nesse exemplo, o pacote se destina a uma LAN 802.3 conectada à ponte; portanto, ele segue seu caminho pelo lado 802.3 da ponte e

prossegue na rede Ethernet. Observe que uma ponte conectando *k* LANs diferentes terá *k* subcamadas MAC diferentes e *k* camadas físicas diferentes, uma para cada tipo.

[arte: imagem original da p. 320]

[Dísticos]

[1] Host A

Rede Pacote

LLC Pacote

MAC 802.11 Pacote

Física 802.11 Pacote

[2] Ponte

Pacote

802.11 Pacote 802.3 Pacote

802.11 Pacote 802.3 Pacote

802.11 Pacote 802.3 Pacote

LAN sem fio Ethernet

[3] Host B

Pacote

Pacote

802.3 Pacote

802.3 Pacote

[F]Figura 4.40

[FL] Operação de uma ponte de LAN entre redes 802.11 e 802.3

Você poderia pensar ingenuamente que é fácil mover um quadro de uma LAN para outra, mas isso não é verdade. Nesta seção, descreveremos algumas dificuldades que serão encontradas durante a tentativa de construção de uma

ponte entre diversas LANs (e MANs) 802. Vamos nos concentrar nos padrões 802.3, 802.11 e 802.16, mas também existem outros, cada qual com seus problemas específicos.

Para começar, cada uma das LANs utiliza um formato de quadro diferente (veja a Figura 4.41). Diferente do que ocorre entre Ethernet, token bus e token ring, cujas características diferentes se devem a razões históricas e a grandes egos empresariais, aqui as distinções são até certo ponto legítimas. Por exemplo, o campo *Duração* do padrão 802.11 se baseia no protocolo MACAW e não faz sentido na Ethernet. Como resultado, qualquer processo de cópia entre LANs diferentes reformatação, o que ocupa tempo da CPU, exige um novo cálculo do total de verificação e introduz a possibilidade de erros não detectados devido a bits defeituosos na memória da ponte.

[arte: imagem original da p. 321]

[Dísticos]

[1] 802.3 Endereço de destino Endereço de origem Comprimento Dados Preenchimento Total de verificação

[2] 802.11 Controle de quadro Duração Endereço 1 Endereço 2 Endereço 3 Seq. Endereço 4 Dados Total de verificação

[3] 802.16 0 EC Tipo CI EK Comprimento ID de conexão CRC de cabeçalho Dados Total de verificação

[F]Figura 4.41

[FL] Os formatos de quadros do IEEE 802. O desenho não está em escala

Um segundo problema é que as LANs interconectadas não funcionam necessaria-mente na mesma taxa de dados. Quando se encaminha uma longa seqüência de quadros de uma LAN rápida para uma LAN mais lenta, a ponte não consegue se livrar dos quadros tão rapidamente quanto eles chegam. Por exemplo, se uma

Ethernet de gigabit estiver despejando bits em uma LAN 802.11b de 11 Mbps à velocidade máxima, a ponte terá de armazená-los no buffer, esperando não esgotar a memória. Pontes que conectam três ou mais LANs passam por um problema semelhante quando várias LANs estão tentando alimentar a mesma LAN de saída ao mesmo tempo, ainda que todas as LANs funcionem à mesma velocidade.

Um terceiro problema e, potencialmente o mais sério de todos, é que todas as LANs 802 têm um tamanho máximo de quadro diferente. Um problema óbvio surge quando um quadro longo tem de ser encaminhado para uma LAN que não pode aceitá-lo. Dividir o quadro em fragmentos está fora de questão nessa camada. Todos os protocolos supõem que os quadros chegaram ou que eles não chegaram. Não há condição de remontar os quadros em unidades menores. Isso não quer dizer que esses protocolos não possam ser criados. Eles poderiam ser projetados e de fato foram. na realidade, nenhum protocolo de enlace de dados oferece esse recurso, e assim as pontes têm de evitar alterar a carga útil do quadro. Basicamente, não há solução para o problema. Quadros grandes demais para serem encaminhados devem ser descartados, porque questão de transparência.

Outro ponto importante é a segurança. Tanto o 802.11 quanto o 802.16 admitem criptografia na camada de enlace de dados. A Ethernet não admite criptografia. Isso significa que os diversos serviços de criptografia disponíveis para as redes sem fios são perdidos quando o tráfego passa por uma Ethernet. Pior ainda, se uma estação sem fio usar criptografia da camada de enlace de dados, não haverá como descriptografar os dados quando eles chegarem a uma rede Ethernet. Se a estação sem fio não usar criptografia, seu tráfego será exposto no enlace aéreo. De qualquer modo, haverá um problema.

Uma solução para o problema de segurança é usar a criptografia em uma camada

mais alta, mas então a estação 802.11 terá de saber se está se comunicando com outra estação em uma rede 802.11 (indicando o uso da criptografia da camada de enlace de dados) ou não (o que significa não usar criptografia). Forçar a estação a fazer uma escolha destrói a transparência.

Um último ponto importante é a qualidade de serviço. Tanto o 802.11 quanto o 802.16 oferecem esse recurso em diversas formas, o primeiro usando o modo PCF e o outro usando conexões de taxa de bits constante. O padrão Ethernet não tem nenhum conceito de qualidade de serviço, e assim o tráfego de qualquer um das outras redes perderá sua qualidade de serviço ao passar por uma rede Ethernet.

[T3] 4.7.2 Interligação de redes locais

A seção anterior tratou dos problemas encontrados na conexão de duas LANs IEEE 802 diferentes através de uma única ponte. Entretanto, em grandes organizações com muitas LANs, o simples fato de interconectá-las gera uma variedade de questões, mesmo que todas elas sejam redes Ethernet. No caso ideal, deveria ser possível sair e comprar pontes projetadas para o padrão IEEE, inserir os conectores nas pontes e tudo funcionaria perfeitamente, no mesmo instante. Não deveria haver necessidade de nenhuma mudança no hardware, nem de alterações no software, nem de configuração de switches de endereços, nem de baixar tabelas de roteamento ou parâmetros, nada. Bastaria ligar os cabos e ir em frente. Além disso, a operação das LANs existentes não deveria ser afetada pelas pontes. Em outras palavras, as pontes deveriam ser completamente transparentes (invisíveis para todo o hardware e software). Surpreendentemente, isso é de fato possível. Agora, vamos ver como essa mágica é realizada.

Em sua forma mais simples, uma ponte transparente opera no modo promíscuo, aceitando cada quadro transmitido em todas as LANs com às quais está

conectada. Como exemplo, considere a configuração da Figura 4.42. A ponte B1 está conectada às LANs 1 e 2, e a ponte B2 está conectada às LANs 2,3 e 4. Um quadro que chega à ponte B1 da LAN 1 destinado a *A* pode ser imediatamente descartado, pois já está na LAN correta; no entanto, um quadro que chega à LAN 1 com destino a *C* ou *F* deve ser encaminhado.

[arte: imagem original da p. 322]

[Dísticos]

[1] Ponte

A B B1

LAN 1

[2] F G H

LAN 4

C B2 D E

LAN 2 LAN 3

[F]Figura 4.42

[FL] Uma configuração com quatro LANs e duas pontes

Quando um quadro chega, uma ponte tem de decidir se deve descartá-lo ou encaminhá-lo e, nesse último caso, em que LAN vai colocá-lo. Essa decisão é tomada, procurando-se o endereço de destino em uma grande tabela (de hash) localizada na ponte. A tabela pode listar cada destino possível e informar a qual linha de saída (LAN) ele pertence. Por exemplo, a tabela de B2 indicaria que *A* pertence à LAN 2, pois o que B2 precisa saber é em que LAN deve colocar os quadros destinados a *A*. O fato de que posteriormente haverá mais encaminhamentos não é de seu interesse.

Quando as pontes são conectadas pela primeira vez, todas as tabelas de hash estão vazias. Nenhuma das pontes sabe onde estão os destinatários; por isso,

elas usam o algoritmo de inundação: cada quadro de entrada para um destino desconhecido é enviado para todas as LANs às quais a ponte está conectada, com exceção da LAN de que ele veio. Com o passar do tempo, as pontes aprendem onde estão os destinatários, como descreveremos a seguir. A partir do momento em que um destinatário se torna conhecido, os quadros destinados a ele são colocados somente na LAN apropriada e não são mais difundidos para todas as redes.

O algoritmo usado pelas redes transparentes é o de **aprendizado reverso.** Como mencionamos antes, as pontes operam no modo promíscuo; portanto, elas vêem todo quadro enviado em qualquer uma das suas LANs. Examinando o endereço de origem, elas podem descobrir que máquina está acessível em qual LAN. Por exemplo, se a ponte B1 da Figura 4.42 vir um quadro na LAN 2 vindo de *C*, ela saberá que *C* pode ser alcançada através da LAN 2; assim, ela faz uma entrada em sua tabela de hash, indicando que os quadros que vão para *C* devem usar a LAN 2. Qualquer quadro subseqüente endereçado a *C* que chegue na LAN 1 será encaminhado; no entanto, um quadro para *C* que chegue na LAN 2 será descartado.

A topologia pode ser alterada à medida que máquinas e pontes são ativadas, desativadas e deslocadas. Para tratar topologias dinâmicas, sempre que uma entrada de tabela de hash é criada, o tempo de chegada do quadro é indicado na entrada. Sempre que chega um quadro cujo destinatário já esteja na tabela, sua entrada é atualizada com a hora atual. Desse modo, o tempo associado a cada entrada informa a última vez que foi visto um quadro proveniente dessa máquina. Periodicamente, um processo na ponte varre a tabela de hash e expurga todas as entradas que tenham mais de alguns minutos. Dessa forma, se um computador for desconectado de sua LAN, levado para outro lugar no prédio e reconectado nesse outro local, dentro de poucos minutos ele estará de volta à operação

normal, sem qualquer intervenção manual. Esse algoritmo também significa que, se uma máquina estiver inativa por alguns minutos, qualquer tráfego enviado a ela terá de ser difundido por inundação, até que ela mesma envie um quadro em seguida.

O procedimento de roteamento para um quadro de entrada depende da LAN em que ele chega (a LAN de origem) e a LAN em que se localiza o seu destino (a LAN de destino), da forma seguinte:

1. Se a LAN de origem e a LAN de destino forem uma só, o quadro será descartado.
2. Se a LAN de origem e a LAN de destino forem diferentes, o quadro será encaminhado.
3. Se a LAN de destino for desconhecida, o quadro será difundido por inundação.

À medida que cada quadro chegar, esse algoritmo será aplicado. Existem chips VLSI de uso especial que pesquisam e atualizam a entrada na tabela, em alguns microssegundos.

[T3] 4.7.3 Pontes de árvores de amplitude

Para aumentar a confiabilidade, algumas instalações usam duas ou mais pontes em paralelo entre os pares de LANs, como mostra a Figura 4.43. Entretanto essa estratégia também introduz alguns problemas adicionais, porque cria loops na topologia.

Podemos ver um exemplo simples desses problemas observando como um quadro *F* com destino desconhecido é tratado na Figura 4.43. Cada ponte, seguindo as regras normais para tratamento de destinos desconhecidos utiliza o algoritmo de inundação que, nesse exemplo, significa apenas copiar o quadro na LAN 2. Logo após, a ponte 1 vê F_2, um quadro com destino desconhecido, que ela copia para a LAN 1, gerando F_3 (não mostrado). Da mesma forma, a ponte 2 copia

F_1 para a LAN 1, gerando F_4 (também não mostrado). A ponte 1 agora encaminha F_4 e a ponte 2 copia F_3. Esse ciclo continua indefinidamente.

[arte: imagem original da p. 324]

[Dísticos]

[1]Quadro copiado por B1 Quadro copiado por B2

F1 F2

[2]LAN 2

[3]B1 Ponte B2

[4]LAN 1

[5]F

Quadro inicial

[F]Figura 4.43

[FL] Duas pontes transparentes paralelas

A solução para essa dificuldade é estabelecer a comunicação entre as pontes e sobrepor a topologia real com uma árvore de amplitude que alcance cada LAN. Na realidade, algumas conexões potenciais entre as LANs são ignoradas no sentido de construir uma topologia fictícia livre de loops. Por exemplo, na Figura 4.44(a), vemos nove LANs interconectadas por dez pontes. Essa configuração pode ser abstraída em um grafo que tem as LANs como nós. Um arco conecta duas LANs quaisquer que estão conectadas por uma ponte. O grafo pode ser reduzido a uma árvore de amplitude, eliminando os arcos mostrados como linhas pontilhadas na Figura 4.44(b). Com a utilização dessa árvore de amplitude, existe exatamente um caminho de cada LAN para qualquer outra LAN. Quando as pontes entram em acordo em relação à árvore de amplitude, tudo o que é encaminhado entre as LANs segue a árvore de amplitude. Como existe um único caminho de cada origem até cada destino, é impossível haver loops.

Para construir a árvore de amplitude, primeiro as pontes precisam escolher uma ponte a ser usada como raiz da árvore. Elas fazem essa escolha da seguinte forma: cada uma transmite por difusão seu número de série, instalado pelo fabricante e com a garantia de ser exclusivo em todo o mundo. A ponte com o número de série mais baixo se torna a raiz. Em seguida, é construída uma árvore de caminhos mais curtos da raiz até cada ponte e a LAN é construída. Essa árvore é a árvore de amplitude. Se uma ponte ou LAN falhar, uma nova árvore será calculada.

O resultado desse algoritmo é o estabelecimento de um caminho exclusivo de cada LAN para a raiz e, portanto, para todas as outras LANs. Apesar de a árvore permitir o acesso a todas as LANs, nem todas as pontes estão necessariamente presentes na árvore (para evitar loops). Mesmo depois que a árvore de amplitude é estabelecida, o algoritmo continua a funcionar durante a operação normal, com a finalidade de detectar automaticamente mudanças na topologia e atualizar a árvore. O algoritmo distribuído usado para a construção da árvore de amplitude foi inventado por Radia Perlman e é descrito detalhadamente em (Perlman, 2000). Ele foi padronizado no IEEE 802.1D.

[arte: imagem original da p. 325a]

[Dísticos]

[1]Ponte

[2]Ponte que faz parte da árvore de amplitude

[3]Ponte que não faz parte da árvore de amplitude

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 4.44

[FL] (a) LANs interconectadas. (b) Uma árvore de amplitude cobrindo as LANs. As

linhas pontilhadas não fazem parte da árvore de amplitude

[T3] 4.7.4 Pontes remotas

Um uso comum de pontes consiste em conectar duas (ou mais) LANs distantes. Por exemplo, uma empresa poderia ter fábricas em várias cidades, cada uma com sua própria LAN. No caso ideal, todas as LANs devem estar interconectadas, de forma que o sistema completo atue como uma grande LAN.

Esse objetivo pode ser alcançado colocando-se uma ponte em cada LAN e conectando-se as pontes aos pares com linhas ponto a ponto (por exemplo, linhas dedicadas de uma companhia telefônica). Um sistema simples, com três LANs, é ilustrado na Figura 4.45. Os algoritmos de roteamento habituais se aplicam aqui. A forma mais simples de ver isso é considerar as três linhas ponto a ponto como LANs sem hosts. Assim, teremos um sistema normal de seis LANs interconectadas por quatro pontes. Nada do que estudamos até agora indica que uma LAN deve conter hosts.

[arte: ver original p. 325b]

[Dísticos]

[1] Ponte

[2] LAN 1

[3]Linha ponto a ponto

[4]LAN 2

[5]LAN 3

[F]Figura 4.45

[FL] As pontes remotas podem ser usadas para interconectar LANs distantes

Vários protocolos podem ser usados nas linhas ponto a ponto. Uma possibilidade é escolher algum protocolo de enlace de dados ponto a ponto padrão como o PPP,

inserindo quadros MAC completos no campo de carga útil. Essa estratégia funciona melhor quando todas as LANs são idênticas, e o único problema é obter quadros para a LAN correta. Outra opção é extrair o cabeçalho MAC e o final na ponte de origem e inserir o restante no campo de carga útil do protocolo ponto a ponto. Então, podem ser gerados novos cabeçalhos e finais MAC na ponte de destino. Uma desvantagem dessa abordagem é que o total de verificação que chega ao host de destino não é o que foi calculado pelo host de origem, e assim os erros causados por bits incorretos na memória de uma ponte podem não ser detectados.

[T3] 4.7.5 Repetidores, hubs, pontes, switches, roteadores e gateways

Até agora neste livro, examinamos diversas maneiras de transferir quadros e pacotes de um segmento de cabo para outro. Mencionamos repetidores, pontes, switches, hubs, roteadores e gateways. Todos esses dispositivos são de uso comum, mas diferem em detalhes sutis e não muito sutis. Por existir uma grande quantidade desses dispositivos, deve valer a pena examiná-los em conjunto para vermos quais são as semelhanças e as diferenças entre eles.

Para começar, esses dispositivos operam em camadas diferentes, como ilustra a Figura 4.46(a). A camada é importante, porque diferentes dispositivos utilizam fragmentos de informações diferentes para decidir como realizar a comutação. Em um cenário típico, o usuário gera alguns dados a serem enviados para uma máquina remota. Esses dados são repassados à camada de transporte, que então acrescenta um cabeçalho — por exemplo, um cabeçalho de TCP — e repassa a unidade resultante à camada de rede situada abaixo dela. A camada de rede adiciona seu próprio cabeçalho para formar um pacote da camada de rede, por exemplo, um pacote IP. Na Figura 4.46(b), vemos o pacote IP sombreado em cor cinza. Em seguida, o pacote vai para a camada de enlace de dados, que adiciona

seu próprio cabeçalho e seu total de verificação (CRC) e entrega o quadro resultante à camada física para transmissão, digamos, por uma LAN.

[arte: ver original p. 326]

[Dísticos]

[1]Camada de aplicação

Camada de transporte

Camada de rede

Camada de enlace de dados

Camada física

[2]Gateway de aplicação

Gateway de transporte

Roteador

Ponte, switch

Repetidor, hub

(a)

[3]Pacote (fornecido pela camada de rede)

Cabeçalho de quadro Cabeçalho de pacote Cabeçalho TCP Dados do usuário CRC

Quadro (criado pela camada de enlace de dados)

(b)

[F]Figura 4.46

[FL] (a) Dispositivos presentes em cada camada. (b) Quadros, pacotes e cabeçalhos

Agora, vamos examinar os dispositivos de comutação e ver como eles se relacionam aos pacotes e quadros. Na parte inferior, na camada física, encontramos os repetidores. Esses dispositivos analógicos estão conectados a

dois segmentos de cabo. Um sinal que aparece em um deles é amplificado e colocado no outro. Os repetidores não reconhecem quadros, pacotes ou cabeçalhos, somente volts. Por exemplo, a Ethernet clássica foi projetada para permitir quatro repetidores, a fim de estender o comprimento máximo de cabo de 500 metros para 2500 metros.

Em seguida, temos os hubs. Um hub tem várias linhas de entrada que ele conecta eletricamente. Os quadros que chegam em quaisquer dessas linhas são enviados a todas as outras. Se dois quadros chegarem ao mesmo tempo, eles colidirão, como ocorre em um cabo coaxial. Em outras palavras, o hub inteiro forma um único domínio de colisão. Todas as linhas que chegam a um hub devem operar na mesma velocidade. Os hubs diferem dos repetidores pelo fato de (normalmente) não amplificarem os sinais de entrada e serem projetados para conter várias placas de linha, cada uma com várias entradas, mas as diferenças são pequenas. Como os repetidores, os hubs não examinam o endereços 802 nem os utilizam de forma alguma. Um hub é mostrado na Figura 4.47(a).

[arte: ver original p. 327]

[Dísticos]

[1]A B C D

Hub

E F G H

(a)

[2]A B C D

Host

Ponte

LAN

E F G H

(b)

[3]A B C D

Switch

E F G H

(c)

[F]Figura 4.47

[FL] (a) Um hub. (b) Uma ponte. (c) Um switch

Agora, vamos passar à camada de enlace de dados, onde encontramos pontes e switches. Acabamos de estudar as pontes com certa profundidade. Uma ponte conecta duas ou mais LANs, como mostra a Figura 4.47(b). Quando um quadro chega, o software da ponte extrai o endereço de destino do cabeçalho de quadro e examina uma tabela, com a finalidade de verificar para onde deve enviar o quadro. No caso de uma rede Ethernet, esse endereço é o destino de 48 bits mostrado na Figura 4.17. Como um hub, uma ponte moderna tem placas de linha, em geral para quatro ou oito linhas de entrada de um certo tipo. Uma placa de linha para Ethernet não pode lidar, digamos, com quadros token ring, porque não sabe onde encontrar o endereço de destino no cabeçalho do quadro. Porém, uma ponte pode ter placas de linha para diferentes tipos de redes e diferentes velocidades. Com uma ponte, cada linha é seu próprio domínio de colisão, em contraste com um hub.

Os switches são semelhantes a pontes pelo fato de ambos basearem o roteamento em endereços de quadro. Na verdade, muitas pessoas utilizam os dois termos de forma intercambiável. A principal diferença é que um switch é usado com maior freqüência para conectar computadores individuais, como mostra a Figura 4.47(c). Como conseqüência, quando o host *A* da Figura 4.47(b) quer enviar um quadro para o host *B*, a ponte recebe o quadro, mas simplesmente o descarta. Em contraste, na Figura 4.47(c), o switch deve

encaminhar ativamente o quadro de *A* até *B*, porque não há outro caminho que o quadro possa seguir. Tendo em vista que cada porta do switch normalmente se conecta a um único computador, os switches precisam ter espaço para muito mais placas de linha do que as pontes destinadas a conectar apenas LANs. Cada placa de linha fornece espaço de buffer para os quadros que chegam a suas portas. Como cada porta é seu próprio domínio de colisão, os switches nunca perdem quadros devido a colisões. Porém, se os quadros chegarem com velocidade maior que aquela em que podem ser retransmitidos, o switch poderá ficar sem espaço de buffer e terá de começar a descartar quadros.

Para atenuar um pouco esse problema, os switches modernos começam a encaminhar quadros tão logo recebem o campo de cabeçalho de destino, mas antes de chegar o restante do quadro (desde que a linha de saída esteja disponível, é claro). Esses switches não utilizam a comutação store-and-forward. Às vezes, eles são chamados **switches de corte**. Em geral, todo o corte é tratado em hardware, enquanto tradicionalmente as pontes continham uma CPU real que fazia a comutação store-and-forward em software. Contudo, como todas as pontes e switches modernos contêm circuitos integrados especiais para comutação, a diferença entre um switch e uma ponte é hoje mais uma questão de marketing do que técnica.

Até o momento, vimos repetidores e hubs, que são bastante semelhantes, bem como pontes e switches, que também são bem parecidos. Agora vamos passar para os roteadores, diferentes de todos os dispositivos anteriores. Quando um pacote entra em um roteador, o cabeçalho de quadro e o final são retirados, e o pacote localizado no campo de carga útil do quadro (sombreado na Figura 4.46) é repassado ao software de roteamento. Esse software utiliza o cabeçalho de pacote para escolher uma linha de saída. No caso de um pacote IP, o cabeçalho do pacote conterá um endereço de 32 bits (IPv4) ou de 128 bits (IPv6), mas não um

endereço 802 de 48 bits. O software de roteamento não vê os endereços de quadro e nem mesmo sabe se o pacote veio de uma LAN ou de uma linha ponto a ponto. Estudaremos os roteadores e o roteamento no Capítulo 5.

Subindo até outra camada, encontramos gateways de transporte. Esses dispositivos conectam dois computadores que utilizam diferentes protocolos de transporte orientados a conexões. Por exemplo, suponha que um computador que utiliza o protocolo TCP/IP orientado a conexões precise se comunicar com um computador que utiliza o protocolo de transporte ATM orientado a conexões. O gateway de transporte pode copiar os pacotes de uma conexão para a outra, reformatando-os caso seja necessário.

Finalmente, os gateways de aplicação reconhecem o formato e conteúdo dos dados e convertem mensagens de um formato para outro. Por exemplo, um gateway de correio eletrônico poderia converter mensagens da Internet em mensagens SMS para telefones móveis.

[T3] 4.7.6 LANs virtuais

Quando foram criadas as primeiras redes locais, grossos cabos amarelos se estendiam pelos conduítes de muitos edifícios comerciais. Por onde eles passavam, todo computador era conectado a esses cabos. Com freqüência, havia muitos cabos conectados a um backbone central (como na Figura 4.39) ou a um hub central. Ninguém parava para pensar que computador pertencia a cada LAN. Todas as pessoas que trabalham em escritórios adjacentes tinham seus equipamentos conectados à mesma LAN, quer elas trabalhassem juntas ou não. A geografia superava a lógica.

Com o advento do 10Base-T e dos hubs na década de 1990, tudo isso mudou. A fiação dos edifícios foi trocada (a um custo considerável) para eliminar todos os grossos cabos amarelos e instalar pares trançados, que iam desde cada escritório

até armários centrais de fiação instalados no fim de cada corredor ou em uma sala de máquinas central, como ilustra a Figura 4.48. Se o encarregado da fiação fosse um visionário, eram instalados pares trançados da categoria 5; se ele fosse um avarento, a fiação telefônica existente (da categoria 3) era usada (até ser substituída alguns anos mais tarde quando a Fast Ethernet surgiu).

[arte: ver original p. 329]

[Dísticos]

[1] Corredor

[2] Conduíte

[3] Hub

[4] Switch

[5] Hub

[6] Escritório

[7] Par trançado até um hub

[F]Figura 4.48

[FL] Um edifício com fiação centralizada, utilizando hubs e um switch

Com a Ethernet usando hubs (e, mais tarde, switches), muitas vezes era possível configurar LANs logicamente, e não fisicamente. Se uma empresa quisesse *k* LANs, bastava adquirir *k* hubs. Pela escolha cuidadosa dos conectores que seriam inseridos em cada hub, os ocupantes de uma LAN podiam ser selecionados de uma forma que fizesse sentido para a organização, sem levar em conta a geografia. É claro que, se duas pessoas do mesmo departamento trabalhassem em edifícios diferentes, elas provavelmente teriam suas máquinas conectadas a hubs diferentes, e portanto a LANs diferentes. Apesar disso, essa é uma situação muito melhor do que distribuir os membros das LANs com base apenas na geografia.

É importante saber quem está conectado a cada LAN? (Afinal, em quase todas as organizações, todas as LANs estão interconectadas.) A resposta é sim, com freqüência isso é importante. Os administradores de redes gostam de agrupar os usuários em LANs de modo a refletir a estrutura organizacional, em lugar do layout físico do edifício, por várias razões. Uma questão é a segurança. Qualquer interface de rede pode ser colocada em modo promíscuo, copiando todo o tráfego que desce pelo canal. Muitos departamentos, como os de pesquisa, patentes e contabilidade, têm informações que não devem sair de seus limites. Em tal situação, faz sentido colocar todas as pessoas de um departamento em uma única LAN, e não permitir que qualquer parte desse tráfego saia da LAN. A gerência não gosta de ouvir que tal distribuição é impossível, a menos que todo o pessoal de cada departamento esteja localizado em escritórios adjacentes, sem intrusos.

Uma segunda questão é a carga. Algumas LANs são utilizadas mais intensamente que outras, e às vezes talvez seja interessante separá-las. Por exemplo, se o pessoal da área de pesquisa estiver realizando várias experiências e alguma delas sair do controle e saturar a LAN, é bem possível que o pessoal da contabilidade não fique muito entusiasmado por ter de doar uma parte de sua capacidade de computação para ajudar os colegas do outro departamento.

Uma terceira questão é a difusão. A maioria das LANs admite a difusão, e muitos protocolos de camadas superiores utilizam intensamente esse recurso. Por exemplo, quando um usuário quer enviar um pacote a um endereço IP representado por *x*, como saber qual endereço MAC deve colocar no quadro? Estudaremos essa questão no Capítulo 5 mas, em resumo, o usuário transmitirá um quadro contendo a seguinte pergunta: A quem pertence o endereço IP *x*? Em seguida, o usuário aguardará uma resposta. Existem muitos outros exemplos de utilização da difusão. À medida que são interconectadas mais LANs, o número de

transmissões por difusão que passam por cada máquina tende a aumentar linearmente com o número de máquinas.

Um problema relacionado à difusão é que, de vez em quando, uma interface de rede sofrerá uma pane e começará a gerar o fluxo infinito de quadros de difusão. O resultado dessa **tempestade de difusão** é que (1) a capacidade da LAN inteira será ocupada por esses quadros e (2) todas as máquinas em todas as LANs interconectadas serão danificadas, processando e descartando todos os quadros que estiverem sendo difundidos.

A princípio, poderia parecer que seria possível limitar o escopo das tempestades de difusão separando as LANs com pontes ou switches; porém, se o objetivo é conseguir transparência (isto é, poder mover uma máquina para uma LAN diferente pela ponte sem que alguém note a mudança), então as pontes têm de encaminhar quadros de difusão.

Depois de verificarmos por que seria interessante para as empresas terem várias LANs com escopo restrito, vamos voltar ao problema de desacoplar a topologia lógica da topologia física. Suponha que dentro da empresa, um usuário seja deslocado de um departamento para outro sem mudar de escritório, mude de escritório sem mudar de departamento. Usando a fiação com hubs, mover o usuário para a LAN correta significa fazer o administrador de rede caminhar até o armário de fiação e puxar o conector correspondente à máquina do usuário de um hub e inseri-lo em um novo hub.

Em muitas empresas, as mudanças organizacionais ocorrem o tempo todo; isso significa que os administradores de sistemas passam muito tempo retirando plugues e inserindo-os de novo em algum outro lugar. Além disso, em alguns casos, a mudança não pode ser feita de modo algum, porque o par trançado da máquina do usuário está longe demais do hub correto (por exemplo, em outro edifício).

Em resposta à solicitação de usuários que desejam maior flexibilidade, os fornecedores de redes começaram a buscar um meio de recompor a fiação dos edifícios inteiramente em software. O conceito resultante é chamado **VLAN (Virtual LAN)** e foi até mesmo padronizado pelo comitê 802. Atualmente, ele está sendo desenvolvido em muitas organizações. Agora, vamos estudá-lo. Para obter informações adicionais sobre VLANs, consulte (Breyer e Riley, 1999; e Seifert, 2000).

As VLANs se baseiam em switches especialmente projetados para reconhecer VLANs, embora também possam ter alguns hubs na periferia, como na Figura 4.48. Para configurar uma rede baseada em VLAN, o administrador da rede decide quantas VLANs haverá, quais computadores estarão em cada VLAN e qual será o nome de cada VLAN. Com freqüência, as VLANs são identificadas (informalmente) por cores, pois assim é possível imprimir diagramas de cores mostrando o layout físico das máquinas, com os membros da LAN vermelha em vermelho, os membros da LAN verde em verde e assim por diante. Desse modo, os layouts físico e lógico são visíveis em um único diagrama.

Como exemplo, considere as quatro LANs da Figura 4.49(a), em que oito das máquinas pertencem à VLAN G (gray — cinza) e sete pertencem à VLAN W (white — branco). As quatro LANs físicas estão conectadas por duas pontes, *B*1 e *B*2. Se fosse usada criação de par trançado centralizada, também poderia haver quatro hubs (não mostrados) mas, logicamente, um cabo multiponto e um hub são idênticos. Um diagrama desse tipo torna a figura um pouco menos confusa. Além disso, o termo "ponte" tende a ser usado hoje principalmente quando existem várias máquinas em cada porta, como nessa figura; porém, em caso contrário, os termos "ponte" e "switch" são essencialmente intercambiáveis. A Figura 4.49(b) mostra as mesmas máquinas e as mesmas VLANs usando switches com um único computador em cada porta.

[arte: ver original p. 331]

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 4.49

[FL] (a) Quatro LANs físicas organizadas por duas pontes em duas VLANs, cinza e branca. (b) As mesmas 15 máquinas organizadas por switches em duas VLANs

Para fazer as VLANs funcionarem corretamente, é necessário definir tabelas de configuração nas pontes ou nos switches. Essas tabelas informam quais são as VLANs acessíveis através de cada uma das portas (linhas). Quando um quadro chega, digamos, da VLAN cinza, ele tem de ser encaminhado para todas as portas marcadas com G. Isso é válido para o tráfego comum (isto é, de unidifusão ou unicast), bem como para tráfego de multidifusão (multicast) e de difusão (broadcast).

Observe que uma porta pode ser identificada com várias cores de VLANs. Vemos isso com maior clareza na Figura 4.49(a). Suponha que a máquina *A* faça a difusão de um quadro. A ponte *B*1 recebe o quadro e observa que ele veio de uma máquina da VLAN cinza, e assim o encaminha em todas as portas identificadas por G (exceto a porta de entrada). Tendo em vista que *B*1 só tem duas outras portas e ambas estão identificadas por G, o quadro é enviado para as duas portas.

Em *B*2, a história é diferente. Aqui, a ponte sabe que não existe nenhuma máquina cinza na LAN 4, e então o quadro não é encaminhado para lá. Ele só vai para a LAN 2. Se um dos usuários da LAN 4 mudar de departamento e for transferido para a VLAN cinza, as tabelas internas de *B*2 terão de ser atualizadas para alterar a identificação dessa porta para GW, em vez de W. Se a máquina *F* for

para a VLAN cinza, a porta para a LAN 2 terá de ser alterada para G, em vez de GW.

Agora, vamos imaginar que todas as máquinas na LAN 2 e na LAN 4 mudem para a cor cinza. Então, não apenas as portas de *B*2 para a LAN 2 e a LAN 4 serão marcadas com G, mas a porta de *B*1 para *B*2 também terá de mudar de GW para G, pois os quadros brancos que chegarem a *B*1 das LANs 1 e 3 não terão mais de ser encaminhados para *B*2. Na Figura 4.49(b), ocorrerá a mesma situação mas, nesse caso, todas as portas que levam a uma única máquina serão identificadas com uma única cor, porque só haverá saída para uma VLAN.

Até agora, partimos do princípio de que, de algum modo, as pontes e os switches sabem qual é a cor de quadro recebido. Como esses dispositivos têm tal conhecimento? São usados três métodos:

1. Toda porta recebe a atribuição de uma cor de VLAN.
2. Todo endereço MAC recebe a atribuição de uma cor de VLAN.
3. Todo protocolo da camada 3 ou endereço IP recebe a atribuição de uma cor de VLAN.

No primeiro método, cada porta é identificada com uma cor de VLAN. Porém, esse método só funcionará se todas as máquinas conectadas a uma porta pertencerem à mesma VLAN. Na Figura 4.49(a), essa propriedade é válida para *B*1 na porta para a LAN 3, mas não na porta para a LAN 1.

No segundo método, a ponte ou o switch tem uma tabela listando o endereço MAC de 48 bits de cada máquina conectada ao dispositivo, juntamente com a VLAN em que essa máquina está. Sob essas condições, é possível misturar VLANs em uma LAN física, como na LAN 1 da Figura 4.49(a). Quando um quadro chega, tudo que a ponte ou o switch tem a fazer é extrair o endereço MAC e procurar em uma tabela para ver de qual VLAN o quadro veio.

No terceiro método, a ponte ou o switch examina o campo de carga útil do

quadro, por exemplo, para classificar todas as máquinas IP como pertencentes a uma VLAN e todas as máquinas AppleTalk como pertencentes a outra VLAN. No primeiro caso, o endereço IP também pode ser usado para identificar a máquina. Essa estratégia é mais útil quando muitas máquinas são notebooks que podem ser encaixados em vários lugares diferentes. Como cada estação de encaixe tem seu próprio endereço MAC, apenas saber qual estação de encaixe foi usada não diz nada sobre a VLAN em que o notebook está.

O único problema com essa abordagem é que ela viola a regra mais fundamental das redes: a independência das camadas. Não interessa à camada de enlace de dados o que está no campo de carga útil. Ela não deve examinar a carga útil e certamente não deve tomar decisões com base no conteúdo desse campo. Uma conseqüência do uso dessa abordagem é que uma mudança no protocolo da camada 3 (por exemplo, uma atualização de IPv4 para IPv6) causa a falha repentina dos switches. Infelizmente, existem no mercado switches que funcionam dessa maneira.

É claro que não há nada de errado no roteamento baseado em endereços IP — quase todo o Capítulo 5 é dedicado ao roteamento IP — mas misturar as camadas é procurar problemas. Um vendedor de switches poderia fazer pouco caso desse argumento, dizendo que seus switches reconhecem tanto o IPv4 quanto o IPv6, e portanto está tudo bem. No entanto, o que acontecerá quando surgir o IPv7? Provavelmente, o vendedor dirá: Ora, por que você não compra novos switches?

[T4] O padrão IEEE 802.1Q

Um pouco mais de reflexão sobre esse assunto revela que o importante é a VLAN do próprio quadro e não a VLAN da máquina transmissora. Se houvesse algum modo de identificar a VLAN no cabeçalho do quadro, a necessidade de inspecionar a carga útil desapareceria. No caso de nova LAN, como a 802.11 ou a

802.16, seria fácil simplesmente acrescentar um campo VLAN no cabeçalho. De fato, o campo *Identificador de conexão* no padrão 802.16 é de certa forma semelhante em espírito a um identificador de VLAN. No entanto, o que fazer no caso do padrão Ethernet, que é a LAN dominante e que não tem nenhum campo sobressalente que possa ser usado como identificador da VLAN?

O comitê 802 do IEEE enfrentou esse problema em 1995. Depois de muita discussão, ele fez o impensável e mudou o cabeçalho do padrão Ethernet. O novo formato foi publicado no padrão **802.1Q** do IEEE, emitido em 1998. O novo formato contém uma tag de VLAN; vamos examiná-lo rapidamente. Não surpreende que a mudança de algo tão bem estabelecido quanto o cabeçalho Ethernet não seja inteiramente trivial. Algumas questões que surgem são:

1. Precisaremos jogar fora várias centenas de milhões de placas Ethernet existentes?
2. Se não, quem irá gerar os novos campos?
3. O que acontecerá com os quadros que já têm o tamanho máximo?

É claro que o comitê 802 estava (ainda que de forma muito dolorosa) consciente desses problemas e teve de apresentar soluções.

A chave para a solução é perceber que os campos VLAN só são realmente usados pelas pontes e switches, e não pelas máquinas dos usuários. Desse modo, na Figura 4.49, não é realmente essencial que eles estejam presentes nas linhas que saem para as estações finais, desde que estejam na linha entre as pontes ou os switches. Portanto, para usar VLANs, as pontes ou os switches têm de estar conscientes da VLAN, mas isso já era uma exigência. Agora, só estamos introduzindo o requisito adicional de que eles devem reconhecer o 802.1Q, o que já acontece no caso dos novos dispositivos.

Quanto a descartar todas as placas Ethernet existente, a resposta é não. Lembre-se de que o comitê 802.3 não poderia nem mesmo fazer as pessoas

transformarem o campo *Tipo* em um campo *Comprimento*. Você pode imaginar a reação ao anúncio de que todas as placas Ethernet existentes teriam de ser jogadas fora. Porém, à medida que novas placas Ethernet entrarem no mercado, espera-se que elas sejam compatíveis com o 802.1Q e preencha corretamente os campos VLAN.

Então, se a origem não gerar os campos VLAN, quem o fará? A resposta é que a primeira ponte ou o primeiro switch capaz de reconhecer a VLAN que tocar um quadro incluirá esses campos, e o último dispositivo do percurso os removerá. Porém, como saber à qual VLAN pertence cada quadro? Bem, a primeira ponte ou switch poderia atribuir um número de VLAN a uma porta, examinar o endereço MAC ou (Deus nos livre) examinar a carga útil. Até as placas Ethernet serem todas compatíveis com o 802.1Q, estaremos de volta ao ponto de partida. Esperamos que todas as placas Ethernet de gigabit sejam compatíveis com o padrão 802.1Q e que, à medida que as pessoas fizerem a atualização para a Ethernet de gigabit, o 802.1Q seja introduzido automaticamente. Quanto ao problema de quadros maiores que 1518 bytes, o 802.1Q simplesmente elevou o limite para 1522 bytes.

Durante o processo de transição, muitas instalações terão as mesmas máquinas antigas (em geral com Ethernet clássica ou Fast Ethernet) que não reconhecem as VLANs e outras (normalmente Ethernet de gigabit) que as reconhecem. Essa situação é ilustrada na Figura 4.50, onde os símbolos sombreados são máquinas que reconhecem VLANs e os símbolos vazios não as reconhecem. Por simplicidade, supomos que todos os switches reconhecem VLANs. Se não for esse o caso, o primeiro switch que reconhecer uma VLAN poderá adicionar as tags com base em endereços MAC ou IP.

[arte: ver original p. 334]

[Dísticos]

[1] Domínio final que reconhece VLANs

[2] Domínio de núcleo que reconhece VLANs

[3] Domínio final de tecnologia antiga

[4] PC de tecnologia antiga

[5] Quadro antigo

[6] Quadro marcado

Comutação feita com o uso de tags

[7] Quadro marcado

Switch que reconhece VLANs

[8] PC que reconhece VLANs

[F]Figura 4.50

[FL] Transição da Ethernet clássica para a Ethernet capaz de reconhecer uma VLAN. Os símbolos sombreados são máquinas conscientes de VLANs, ao contrário dos símbolos vazios

Nessa figura, as placas Ethernet que reconhecem VLANs geram quadros marcados (isto é, 802.1Q) diretamente, e a comutação adicional utiliza essas tags. Para realizar essa comutação, os switches têm de saber quais VLANs são acessíveis em cada porta, da mesma forma que antes. Saber que um quadro pertence à VLAN cinza não ajuda muito enquanto o switch não souber quais portas se conectam a máquinas da VLAN cinza. Desse modo, o switch precisa de uma tabela indexada por VLAN, informando quais portas usar e se elas reconhecem VLANs ou são antigas.

Quando um PC de tecnologia antiga envia um quadro a um switch que reconhece VLANs, o switch constrói um novo quadro marcado com base em seu conhecimento da VLAN do transmissor (usando a porta, o endereço MAC ou o endereço IP). Desse momento em diante, não importa mais se o transmissor tinha uma máquina antiga. De modo semelhante, um switch que precisa entregar um

quadro marcado a uma máquina antiga tem de reformatar o quadro no formato antigo antes de entregá-lo.

Agora, vamos examinar o formato de quadro 802.1Q. Ele está representado na Figura 4.51. A única mudança é a adição de um par de campos de 2 bytes. O primeiro é o campo *ID de protocolo de VLAN*, que sempre tem o valor 0x8100. Tendo em vista que esse número é maior que 1500, todas as placas Ethernet o interpretam como um tipo, e não como um comprimento. O que uma placa antiga faz com um quadro desse tipo é discutível, pois tais quadros não devem ser enviados a placas antigas.

[arte: ver original p. 335]

[Dísticos]

[1]802.3 Endereço de destino Endereço de origem Comprimento Dados Preenchimento Total de verificação

[2]802.1Q Endereço de destino Endereço de origem Tag Comprimento Dados Preenchimento Total de verificação

[3]ID de protocolo de VLAN (0x8100)

[4]Pri CFI Identificador de VLAN

[F]Figura 4.51

[FL] Os formatos de quadros Ethernet 802.3 (antigo) e 802.1Q

O segundo campo de 2 bytes contém três subcampos. O principal é o *Identificador de VLAN*, que ocupa os 12 bits de baixa ordem. É isso que interessa — a que VLAN o quadro pertence? O campo de 3 bits *Prioridade* não tem nenhuma relação com VLANs mas, como a mudança no cabeçalho Ethernet é um evento que acontece uma vez em cada década, demora três anos e envolve uma centena de pessoas, por que não incluir alguns outros benefícios? Esse campo torna possível distinguir o tráfego de tempo real permanente do tráfego de tempo

real provisório e do tráfego não relacionado ao tempo, a fim de fornecer melhor qualidade de serviço em redes Ethernet. Ele é necessário para voz sobre a Ethernet (embora o IP tivesse um campo semelhante durante um quarto de um século sem que ninguém jamais o tenha usado).

O último bit, *CFI (Canonical Format Indicator — indicador de formato canônico)*, deveria ter sido chamado *CEI (Corporate Ego Indicator — indicador de ego corporativo)*. Originalmente, ele foi criado para indicar endereços MAC little-endian *versus* endereços MAC big-endian, mas esse uso se perdeu em outras controvérsias. Sua presença agora indica que a carga útil contém um quadro 802.5 congelado que está esperando encontrar outra LAN 802.5 no destino, enquanto está sendo transportado por uma rede Ethernet nesse meio tempo. É claro que toda essa organização não tem nenhuma relação com as VLANs. No entanto, a política do comitê de padrões não é diferente da política comum: se você votar a favor do meu bit, eu votarei a favor do seu.

Como mencionamos antes, quando um quadro marcado chega a um switch que reconhece VLANs, o switch utiliza a ID da VLAN como um índice em uma tabela, para descobrir através de que portas deve enviar o quadro. Porém, de onde vem a tabela? Se ela for construída manualmente, voltaremos à estaca zero: a configuração manual de pontes. A beleza da ponte transparente é o fato de ela ser plug-and-play e não exigir qualquer configuração manual. Seria uma vergonha terrível perder essa propriedade. Felizmente, as pontes que reconhecem VLANs também podem se autoconfigurar com base na observação das tags que passam por elas. Se um quadro marcado como VLAN 4 chegar na porta 3, então aparentemente alguma máquina na porta 3 está na VLAN 4. O padrão 802.1Q explica como construir as tabelas dinamicamente, em grande parte por meio de referências a porções apropriadas do algoritmo de Perlman padronizado no 802.1D.

Antes de encerrarmos o assunto de roteamento de VLANs, vale a pena fazermos uma última observação. Muitas pessoas no universo da Internet e das redes Ethernet defendem de forma enfática a interligação de redes sem conexões e se opõem violentamente a qualquer sinal de conexões na camada de enlace de dados ou de rede. Ainda assim, as VLANs introduzem algo surpreendentemente semelhante a uma conexão. Para usar as VLANs de forma apropriada, cada quadro transporta um novo identificador especial que é usado como um índice para uma tabela interna do switch, a fim de procurar o local para onde o quadro deve ser enviado. É isso mesmo o que acontece nas redes orientadas a conexões. Nas redes sem conexões, deve-se utilizar o endereço de destino para roteamento e não alguma espécie de identificador de conexão. Veremos mais detalhes sobre essa rejeição às conexões no Capítulo 5.

[T2] 4.8 Resumo

Algumas redes administram todo o fluxo de comunicações através de um único canal. Nessas redes, a grande questão é a alocação desse canal entre as estações que desejam utilizá-lo. Foram criados diversos algoritmos de alocação de canais. A Figura 4.52 contém um resumo dos métodos de alocação de canais mais importantes.

Os esquemas de alocação mais simples são FDM e TDM. Eles são eficientes quando o número de estações é pequeno e fixo, e quando o tráfego é continuo. Ambos são amplamente utilizados nessas circunstâncias como, por exemplo, para dividir a largura de banda nos enlaces usados como troncos telefônicos.

No entanto, a FDM e a TDM não são boas opções quando o número de estações é grande e variável, ou quando o tráfego ocorre em rajadas. O protocolo ALOHA, com e sem segmentação (ALOHA puro ou slotted ALOHA), foi proposto como uma alternativa. O ALOHA e suas numerosas variantes e derivados foram amplamente

discutidos, analisados e usados em sistemas reais.

Quando é possível detectar o estado do canal, uma estação pode evitar iniciar uma transmissão enquanto outra estação está transmitindo. A técnica, chamada de detecção de portadora, levou a uma série de protocolos que podem ser usados em LANs e MANs.

[arte: ver original p. 337]

[Tabela]

Método	**Descrição**
FDM	Dedica uma banda de freqüência a cada estação
WDM	Um esquema de FDM dinâmico para fibra
TDM	Dedica um slot de tempo a cada estação
ALOHA puro	Transmissão não sincronizada em qualquer momento
Slotted ALOHA	Transmissão aleatória em slots de tempo bem definidos
CSMA 1-persistente	CSMA padrão
CSMA não persistente	Retardo aleatório quando o canal é detectado como ocupado
CSMA P-persistente	CSMA, mas com a probabilidade p de persistência
CSMA/CD	CSMA, mas cancela a operação ao detectar uma colisão
Mapa de bits	Escalonamento em rodízio, usando um mapa de bits
Contagem regressiva binária	A estação pronta com o número mais alto transmite em seguida
Percurso em árvore	Disputa reduzida por ativação seletiva
MACA, MACAW	Protocolos de LANs sem fios
Ethernet	CSMA/CD com recuo binário exponencial
FHSS	Espectro de dispersão de saltos de freqüência
DSSS	Espectro de dispersão de seqüência direta
CSMA/CA	CSMA com abstenção de colisão

[F]Figura 4.52

[FL] Métodos e sistemas de alocação de canais para um canal comum

É bem conhecida uma classe de protocolos que elimina por completo a disputa, ou pelo menos a reduz consideravelmente. A contagem regressiva binária elimina totalmente a disputa. O protocolo de percurso em árvore reduz a disputa dividindo de forma dinâmica as estações em dois grupos disjuntos, dos quais apenas um pode transmitir. Ele tenta fazer a divisão de um modo tal que apenas uma estação que esteja pronta para transmitir tenha permissão para fazê-lo.

As LANs sem fios têm seus próprios problemas e soluções. O maior problema é provocado pelas estações ocultas, que fazem com que o CSMA não funcione. Uma classe de soluções, tipificada pelo MACA e pelo MACAW, tenta estimular transmissões em torno do destino, a fim de melhorar o funcionamento do CSMA. O espectro de dispersão de saltos de freqüência e o espectro de dispersão de seqüência direta também são usados. O IEEE 802.11 combina o CSMA e o MACAW para produzir o CSMA/CA.

A Ethernet é a forma dominante de interligação de redes locais. Ela utiliza o CSMA/CD para a alocação de canais. As versões mais antigas usavam um cabo que se estendia de uma máquina para outra, mas agora pares trançados para hubs e switches são mais comuns. As velocidades aumentaram de 10 Mbps para 1 Gbps e ainda estão aumentando.

As LANs sem fios estão se tornando comuns, com o padrão 802.11 dominando o mercado. Sua camada física permite cinco modos diferentes de transmissão, incluindo infravermelho, diversos esquemas de espectros de dispersão e um sistema FDM multicanal. Ele pode operar com uma estação base em cada célula, mas também pode operar sem uma estação base. O protocolo é uma variante do MACAW, com detecção de portadora virtual.

As MANs sem fios estão começando a surgir. Elas são sistemas de banda larga que usam ondas de rádio para substituir a parte final das conexões telefônicas. São usadas as técnicas tradicionais de modulação de banda estreita. A qualidade de serviço é importante, com o padrão 802.16 definindo quatro classes (taxa de bits constante, duas classes de taxa de bits variável e uma classe de melhor esforço).

O sistema Bluetooth também é sem fios, mas se destina ao mercado de desktop, para conectar fones de ouvido e outros periféricos a computadores sem a utilização de fios. Ele também se destina a conectar periféricos, como equipamentos de fax, a telefones móveis. Da mesma forma que o 802.11, ele utiliza o espectro de dispersão de saltos de freqüência na banda ISM. Devido ao nível de ruído esperado de muitos ambientes e à necessidade de interação em tempo real, foi construído um sistema elaborado de correção antecipada de erros em seus diversos protocolos.

Com tantas LANs diferentes, é necessário um modo para interconectar todas elas. As pontes e os switches são usados para esse fim. O algoritmo de árvore de amplitude é utilizado para elaborar pontes plug-and-play. Um novo desenvolvimento no campo da interconexão de LANs é a VLAN, que separa a topologia lógica das LANs de sua topologia física. Um novo formato para quadros Ethernet (802.1Q) foi introduzido com o objetivo de facilitar o uso de VLANs nas organizações.

[T2] Problemas

1. Para resolver este problema, use uma fórmula deste capítulo, mas primeiro enuncie a fórmula. Os quadros chegam aleatoriamente em um canal de 100 Mbps para transmissão. Se estiver ocupado quando um quadro chegar, o canal aguardará sua vez em uma fila. O comprimento do quadro está distribuído

exponencialmente com uma média de 10.000 bits/quadro. Para cada uma das taxas de chegada de quadros a seguir, determine o retardo experimentado pelo quadro médio, incluindo o tempo de enfileiramento e o tempo de transmissão.

(a) 90 quadros/s.

(b) 900 quadros/s.

(c) 9000 quadros/s.

2. Um grupo de *N* estações compartilha um canal ALOHA puro de 56 kbps. Cada estação transmite em média um quadro de 1.000 bits a cada 100 s, mesmo que o anterior ainda não tenha sido enviado (as estações podem, por exemplo, armazenar em buffer os quadros enviados). Qual é o valor máximo de *N*?

3. Compare o retardo do ALOHA puro com o do slotted ALOHA com uma carga baixa. Qual deles é menor? Explique sua resposta.

4. Dezenas de milhares de estações de reservas aéreas estão disputando o uso de um único canal slotted ALOHA. A estação média faz 18 solicitações/hora. Um slot tem 125 μs. Qual é a carga total aproximada do canal?

5. Uma grande população de usuários do ALOHA tenta gerar 50 solicitações/s, incluindo os quadros originais e as retransmissões. O tempo é dividido em unidades de 40 ms.

(a) Qual é a chance de sucesso na primeira tentativa?

(b) Qual é a probabilidade de haver exatamente *k* colisões antes de se obter sucesso?

(c) Qual é o número esperado de tentativas de transmissão necessárias?

6. A medição de um canal slotted ALOHA com um número infinito de usuários mostra que 10% dos slots estão ociosos.

(a) Qual é a carga do canal, representada por *G*?

(b) Qual é o throughput?

(c) O canal está sobrecarregado ou subutilizado?

7. Em um sistema slotted ALOHA com uma população infinita, o número médio de slots que uma estação aguarda entre uma colisão e sua retransmissão é 4. Represente em um diagrama a curva de variação do retardo com o throughput desse sistema.

8. Quanto tempo uma estação *s* terá de esperar, na pior das hipóteses, antes de poder começar a transmitir seu quadro sobre uma LAN que use:

(a) o protocolo de mapa de bits básico?

(b) o protocolo de Mok e Ward com permuta de números de estações virtuais?

9. Uma LAN utiliza a versão de Mok e Ward da contagem regressiva binária. A certa altura, as 10 estações são identificadas pelos números de estação virtual 8, 2, 4, 5, 1, 7, 3, 6, 9 e 0. As três estações que devem transmitir em seguida são 4, 3 e 9, nessa ordem. Quais serão os novos números de estação virtual depois que todas três tiverem concluído as suas transmissões?

10. Dezesseis estações, numeradas de 1 a 16, estão disputando o uso de um canal compartilhado que emprega o protocolo de percurso em árvore adaptativo. Se todas as estações cujos endereços são números primos de repente ficarem disponíveis ao mesmo tempo, quantos slots de bits serão necessários para resolver a disputa?

11. Um conjunto de 2^n estações usa o protocolo de percurso em árvore adaptativo para arbitrar o acesso a um cabo compartilhado. Em um determinado instante, duas delas se tornam disponíveis. Qual será o número máximo, mínimo e médio de slots do percurso em árvore, se 2^n for muito maior que 1?

12. As LANs sem fios que estudamos usavam protocolos como MACA em vez de utilizarem o CSMA/CD. Em que condições, se houver, seria possível usar o CSMA/CD?

13. Que propriedades os protocolos de acesso ao canal WDMA e GSM têm em comum? Veja detalhes sobre o GSM no Capítulo 2.

14. Seis estações, de *A* até *F*, se comunicam usando o protocolo MACA. Seria possível duas transmissões ocorrerem simultaneamente? Explique sua resposta.

15. Um prédio comercial de sete andares tem 15 escritórios adjacentes por andar. Cada escritório contém uma tomada (um soquete) para um terminal na parede frontal. Dessa forma, as tomadas formam uma grade retangular em um plano vertical, com uma distância de 4 m entre as tomadas, tanto no sentido horizontal quanto no vertical. Partindo do princípio de que é possível passar um cabo linear entre qualquer par de tomadas, seja no sentido horizontal, vertical ou diagonal, quantos metros de cabo seriam necessários para conectar todas as tomadas usando:

(a) uma configuração em estrela com um único roteador no centro?

(b) uma LAN 802.3?

16. Qual é a taxa de baud da rede Ethernet padrão de 10 Mbps?

17. Estruture a codificação Manchester do fluxo de bits: 0001110101.

18. Estruture a codificação Manchester diferencial correspondente ao fluxo de bits do problema anterior. Parta do princípio de que a linha está inicialmente no estado baixo.

19. Uma LAN CSMA/CD de 10 Mbps (não 802.3) com a extensão de 1 km tem uma velocidade de propagação de 200 m/µs. Não são permitidos repetidores nesse sistema. Os quadros de dados têm 256 bits, incluindo 32 bits de cabeçalho, totais de verificação e outras formas de overhead. O primeiro slot de bits depois de uma transmissão bem-sucedida é reservado para o receptor capturar o canal com o objetivo de enviar um quadro de confirmação de 32 bits. Qual será a taxa de dados efetiva, excluindo o overhead, se partirmos do princípio de que não há colisões?

20. Duas estações CSMA/CD estão tentando transmitir arquivos longos (de vários quadros). Depois que cada quadro é enviado, elas disputam o canal usando o

algoritmo de recuo binário exponencial. Qual é a probabilidade de a disputa terminar na rodada de número *k*, e qual é o número médio de rodadas por período de disputa?

21. Considere a construção de uma rede CSMA/CD que funciona a 1 Gbps sobre um cabo de 1 km, sem repetidores. A velocidade do sinal no cabo é 200.000 km/s. Qual é o tamanho mínimo de quadro?

22. Um pacote IP a ser transmitido por uma rede Ethernet tem 60 bytes de comprimento, incluindo todos os seus cabeçalhos. Se o LLC não estiver em uso, será necessário utilizar preenchimento no quadro Ethernet? Em caso afirmativo, de quantos bytes?

23. Os quadros Ethernet devem ter pelo menos 64 bytes para garantir que o transmissor ainda estará ativo na eventualidade de ocorrer uma colisão na extremidade remota do cabo. O tamanho mínimo de quadro nas redes com cabeamento Fast Ethernet também é de 64 bytes, mas esse cabeamento é capaz de transportar o mesmo número de bits com uma velocidade 10 vezes maior. Como é possível manter o mesmo tamanho mínimo de quadro?

24. Alguns livros citam o tamanho máximo de um quadro Ethernet como 1518 bytes em vez de 1500 bytes. Eles estão errados? Explique sua resposta.

25. A especificação 1000Base-SX estabelece que o clock deverá funcionar a 1250 MHz, embora a Ethernet de gigabit só deva entregar 1 Gbps. Essa velocidade mais alta tem a finalidade de oferecer uma margem extra de segurança? Se não, o que está acontecendo?

26. Quantos quadros por segundo a Ethernet de gigabit podem manipular? Pense cuidadosamente e leve em conta todos os casos relevantes. *Sugestão:* O fato de ela ser uma Ethernet de *gigabit* é importante.

27. Identifique duas redes que permitam que os quadros sejam reunidos em seqüência. Por que essa característica é importante?

28. Na Figura 4.27 são mostradas quatro estações, *A*, *B*, *C* e *D*. Qual das duas últimas estações você acha que está mais próxima de *A*, e por quê?

29. Suponha que uma LAN 802.11b de 11 Mbps esteja transmitindo quadros de 64 bytes em seqüência por um canal de rádio com uma taxa de erros de bits igual a 10^{-7}. Quantos quadros por segundo serão danificados em média?

30. Uma rede 802.16 tem uma largura de canal de 20 MHz. Quantos bits/s podem ser enviados a uma estação de assinante?

31. O IEEE 802.16 admite quatro classes de serviço. Que classe de serviço é a melhor opção para transmitir vídeo não compactado?

32. Apresente duas razões pelas quais as redes poderiam usar um código de correção de erros em vez de detecção de erros e retransmissão.

33. Na Figura 4.35, observamos que um dispositivo Bluetooth pode estar em duas piconets ao mesmo tempo. Existe alguma razão pela qual um dispositivo não possa ser o mestre em ambas as piconets ao mesmo tempo?

34. A Figura 4.25 mostra diversos protocolos da camada física. Qual deles é o mais próximo do protocolo da camada física do Bluetooth? Qual é a maior diferença entre os dois?

35. O Bluetooth admite dois tipos de enlaces entre um mestre e um escravo. Quais são eles e para que cada um é usado?

36. Os quadros de baliza na variante de espectro de dispersão de salto de freqüência do 802.11 contém o tempo de parada. Você acha que os quadros de baliza análogos no Bluetooth também contém o tempo de parada? Explique sua resposta.

37. Considere as LANs interconectados representadas na Figura 4.44. Suponha que os hosts *a* e *b* estejam na LAN 1, que *c* esteja na LAN 2 e que *d* esteja na LAN 8. Inicialmente, as tabelas de hash de todas as pontes estão vazias, sendo suada a árvore de amplitude da Figura 4.44(b). Mostre como as tabelas de hash de

pontes diferentes se alteram depois de ocorrer em seqüência cada um dos eventos a seguir, primeiro (a), depois (b) e assim por diante:

(a) *a* transmite para *d*.

(b) *c* transmite para *a*.

(c) *d* transmite para *c*.

(d) *d* se desloca para a LAN 6.

(e) *d* transmite para *a*.

38. Uma conseqüência Do uso de uma árvore de amplitude para encaminhar quadros em uma LAN estendida é que algumas pontes não podem participar do encaminhamento de quadros. Identifique três dessas pontes na Figura 4.44. Existe alguma razão para manter essas pontes, embora elas não sejam usadas no encaminhamento?

39. Imagine que um switch tenha placas de linha para quatro linhas de entrada. Freqüentemente, um quadro que chega em uma das linhas tem de sair em outra linha na mesma placa. Que decisões o projetista do switch deve tomar como resultado dessa situação?

40. Um switch projetado para uso com Fast Ethernet tem um @@@backplane que pode mover 10 Gbps. Quantos quadros/s ele pode tratar no pior caso?

41. Considere a rede da Figura 4.49(a). Se a máquina *J* repentinamente se tornasse branca, seria necessário alguma alteração no método de identificação? Em caso afirmativo, quais seriam as mudanças?

42. Descreve resumidamente a diferença entre switches store-and-forward e de corte.

43. Os switches store-and-forward levam vantagem sobre os switches de corte no que se refere a quadros danificados. Explique qual é essa vantagem.

44. Para fazer as VLANs funcionarem, são necessárias tabelas de configuração nos switches e nas pontes. E se as VLANs da Figura 4.49(a) usarem hubs em vez

de cabos multiponto? Os hubs também necessitam de tabelas de configuração? Por que ou por que não?

45. Na Figura 4.50, o switch no domínio final de tecnologia antiga do lado direito é um switch que reconhece VLANs. Seria possível usar ali um switch de tecnologia antiga? Nesse caso, como isso funcionaria? Se não, por que não?

46. Escreva um programa que simule o comportamento do protocolo CSMA/CD sobre Ethernet quando existirem *N* estações prontas para transmitir enquanto um quadro está sendo transmitido. Seu programa deve informar os momentos em que cada estação inicia com êxito a transmissão de seu quadro. Suponha que ocorra um pulso de clock em cada período de slot (51,2 microssegundos) e que uma seqüência de detecção de colisão e transmissão de interferência demore um período de slot. Todos os quadros têm o comprimento máximo permitido.

[TA1]Capítulo

[TA2]5

[T1] A camada de rede

A camada de rede está relacionada à transferência de pacotes da origem para o destino. Chegar ao destino pode exigir vários hops (saltos) em roteadores intermediários ao longo do percurso. Essa função contrasta claramente com a função da camada de enlace de dados, que tem o objetivo mais modesto de apenas mover quadros de uma extremidade de um fio até a outra. Portanto, a camada de rede é a camada mais baixa que lida com a transmissão fim a fim. Para atingir seus objetivos, a camada de rede deve conhecer a topologia da sub-rede de comunicações (ou seja, o conjunto de todos os roteadores) e escolher os caminhos mais apropriados através dela. A camada de rede também deve ter o cuidado de escolher rotas que evitem sobrecarregar algumas das linhas de comunicação e roteadores enquanto deixam outras ociosas. Por fim, quando a origem e o destino estão em redes diferentes, ocorrem novos problemas, e cabe à camada de rede lidar com eles. Neste capítulo, estudaremos todas essas questões e as ilustraremos, uando principalmente a Internet e o protocolo de sua camada de rede, o IP, embora também sejam examinadas as redes sem fios.

[T2] 5.1 Questões de projeto da camada de rede

Nas seções a seguir, apresentaremos informações introdutórias sobre algumas das questões com que os projetistas da camada de rede devem se preocupar. Dentre elas, estão o serviço oferecido à camada de transporte e o projeto interno da sub-rede.

[T3] 5.1.1 Comutação de pacotes store-and-forward

Antes de começarmos a explicar os detalhes da camada de rede, vale a pena redefinir o contexto em que operam os protocolos a camada de rede. Esse contexto pode ser visto na Figura 5.1. Os principais componentes do sistema são o equipamento da concessionária de comunicações (roteadores conectados por linhas de transmissão), mostrados na elipse sombreada, e o equipamento dos clientes, mostrado fora da elipse. O host *H*1 está diretamente conectado a um dos roteadores da concessionária de comunicações, denominado *A*, por uma linha dedicada. Em contraste, *H*2 está em uma LAN com um roteador *F* pertencente ao cliente e operado por ele. Esse roteador também tem uma linha dedicada para o equipamento da concessionária de comunicações. Mostramos *F* fora da elipse porque ele não pertence à concessionária de comunicações; porém, em termos de construção, software e protocolos, é bem provável que ele não seja diferente dos roteadores da concessionária de comunicações. O fato de ele pertencer à sub-rede é discutível mas, para os propósitos deste capítulo, roteadores no local do cliente são consierados parte da sub-rede, porque excutam os mesmos algoritmos que os roteadores da concessionária de comunicações (e nossa principal preocupação aqui é o estudo dos algoritmos).

[arte: ver original p. 344]

[Dísticos]

[1] Roteador

[2] Equipamento da concessionária de comunicações

[3] Processo P1

[4] Pacote

[5] Processo P2

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.1

[FL] O ambiente dos protocolos da camada de rede

Esse quipamento é suado da maneira descrita a seguir. Um host com um pacote a enviar o transmite para o roteador mais próximo, seja em sua própria LAN ou sobre um enlace ponto a ponto para a concessionária de comunicações. O pacote é armazenado ali até chegar totalmente, de forma que o total de verificação possa ser conferido. Em seguida, ele é encaminhado para o próximo roteador ao longo do caminho, até alcançar o host de destino, onde é entregue. esse mecanismo é a comutação de pacotes store-and-forward, como vimos em capítulos anteriores.

[T3] 5.1.2 Serviços oferecidos à camada de transporte

A camada de rede oferece serviços à camada de transporte na interface entre a camada de rede e a camada de transporte. Uma questão importante é identificar os tipos de serviços que a camada de rede oferece à camada de transporte. Os serviços da camada de rede foram projetados tendo em vista os objetivos a seguir.

1. Os serviços devem ser independentes da tecnologia de roteadores.
2. A camada de transporte deve ser isolada do número,do tipo e da topologia dos roteadores presentes.
3. Os endereços de rede que se tornaram disponíveis para a camada de transporte devem usar um plano de numeração uniforme, mesmo nas LANs e WANs.

Tendo definido esses objetivos, os projetistas da camada de rede têm muita liberdade para escrever especificações detalhadas dos serviços a serem oferecidos à camada de transporte. Essa liberdade costuma se transformar em uma violenta batalha entre duas facções. A discussão se concentra na seguinte

questão: a camada de rede deve fornecer serviço orientado a conexões ou serviço sem conexões?

Um lado (representado pela comunidade da Internet) alega que a tarefa dos roteadores é tão-somente movimentar pacotes. Na visão dessas pessoas (baseada em 30 anos de experiência com uma rede de computadores ativa e real), a sub-rede é inerentemente pouco confiável, independente de como tenha sido projetada. Portanto, os hosts devem aceitar o fato de que a rede é pouco confiável e fazerem eles próprios o controle de erros (ou seja, detecção e correção de erros) e o controle de fluxo.

Esse ponto de vista leva rapidamente à conclusão de que o serviço de rede deve ser sem conexões, praticamente restrito às primitivas SEND PACKET e RECEIVE PACKET. Em particular, não deve ser realizada nenhuma forma de ordenação de pacotes e controle de fluxo, pois os hosts cuidarão disso de qualquer maneira, e em geral não há grande vantagem em fazer a mesma tarefa duas vezes. Além disso, cada pacote deve ter o endereço de destino completo, pois todos são transportados independentemente de seus predecessores, se for o caso.

O outro lado (representado pelas companhias telefônicas) alega que a sub-rede deve fornecer um serviço orientado a conexões confiável. Elas afirmam que os 100 anos de experiência bem-sucedida com o sistema telefônico mundial servem como um bom guia. De acordo com essa visão, a qualidade de serviço é o fator dominante e, sem conexões na sub-rede, é muito difícil alcançar qualidade de serviço, eme special no caso de tráfego de tempo real, como voz e vídeo.

Essas duas opiniões são melhor exemplificadas pela Internet e pelas redes ATM. A Internet oferece serviço da camada de rede sem conexões; as redes ATM oferecem serviço da camada de rede orientado a conexões. Enretanto, é interessante observar que, à medida que as garantias de qualidade de serviço estão se tornando cada vez mais importantes, a Internet está evoluindo. Em

particular, ela está começcando a adquirir propriedades normalmente associadas ao serviço orientado a conexões, como veremos mais adiante. Na realidade, tivemos a oportunidade de observar um pouco dessa evolução durante nosso estudo das VLANS no Capítulo 4.

[T3] 5.1.3 Implementação do serviço sem conexões

Depois de analisar as duas classes de serviço que a camada de rede pode oferecer a seus usuários, chegou a hora de vermos como essa camada funciona por dentro. São possíveis duas organizações diferentes, dependendo do tipo de serviço oferecido. Se for oferecido o serviço sem conexões, os pacotes serão injetados individualmente na sub-rede e roteados de modo independente uns dos outros. Não será necessária nenhuma configuração antecipada. Nesse contexto, os pacotes freqüentemente são chamadas **datagramas** (em uma analogia com os telegramas) e a sub-rede será denominada **sub-rede de datagramas.** Se for usado o serviço orientado a conexões, terá de ser estabelecido um caminho desde o roteador de origem até o roteador de destino, antes de ser possível enviar quaisquer pacotes de dados. Essa conexão é chamada **circuito virtual**, em analogia com os circuitos físicos estabelecidos pelo sistema telefônico, e a sub-rede é denominada **sub-rede de circuitos vituais.** Nesta seção, examinaremos as sub-redes de datagramas; na próxima, estudaremos as sub-redes de circuitos virtuais.

Vejamos agora como funciona uma sub-rede de datagramas. Suponha que o processo *P*1 da Figura 5.2 tenha uma longa mensagem para *P*2. Ele entrega a mensagem à camada de transporte, com instruções para que ela seja entregue a *P*2 do host *H*2. O código da camada de transporte funciona em *H*1, em geral dentro do sistema operacional. Ele acrescenta um cabeçalho de transporte ao início da mensagem e entrega o resultado à camada de rede, que talvez seja

simplesmente outro procedimento no sistema operacional.

[arte: ver original p. 346]

[Dísticos]

[1] Pacote

[2] Roteador

[3] Equipamento da concessionária de comunicações

[4] Processo P1

[5] Processo P2

[6] Tabela de A

Inicialmente Mais tarde

[7] Tabela de C

[8] Tabela de E

[9] Linha de destino

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.2

[FL] Roteamento em uma sub-rede de datagramas

Vamos supor que a mensagem seja quatro vezes mais longa que o tamanho máximo de pacote, e portanto que a camada de rede tem de dividi-la em quatro pacotes, 1, 2, 3 e 4, e enviar cada um deles ao roteador *A*, usando algum protocolo ponto a ponto como, por exemplo, o PPP. Nesse ponto, a concessionária de comunicações assume o controle. Todo roteador tem uma tabela interna que informa para onde devem ser enviados os pacotes a serem entregues a cada destino possível. Cada entrada da tabela é um par que consiste em um destino e na linha de saída a ser utilizada para esse destino. Somente

podem ser usadas linhas conectadas diretamente. Por exemplo, na Figura 5.2, *A* tem apenas duas linhas de saída — para *D* e *C* — e assim todo pacote recebido deve ser enviado a um desses roteadores, mesmo que o destino final seja algum outro roteador. A tabela de roteamento inicial de *A* é mostrada na figura sob o título "Inicialmente".

À medida que chegaram ao roteador *A*, os pacotes 1, 2 e 3 foram armazenados por algum tempo (para que seus totais de verificação fossem conferidos). Em seguida, cada um deles foi encaminhado para *C*, de acordo com a tabela de *A*. O pacote 1 foi então encaminhado para *E* e depois para *F*. Chegando a *F*, ele foi encapsulado em um quadro da camada de enlace de dados e transmitido para *H*2 pela LAN. Os pacotes 2 e 3 seguiram a mesma rota.

Entretanto, aconteceu algo diferente com o pacote 4. Quando chegou ao roteador *A*, ele foi enviado para o roteador *B*, embora seu destino também fosse *F*. Por alguma razão, *A* decidiu enviar o pacote 4 por uma rota diferente da que foi usada para os três primeiros pacotes. Talvez ele tenha tomado conhecimento de uma obstrução de tráfego em algum lugar no caminho *ACE* e tenha atualizado sua tabela de roteamento, como mostamos na figura sob o título "Mais tarde". O algoritmo que gerencia as tabelas e toma as decisões de roteamento é chamado **algoritmo de roteamento.** Os algoritmos de roteamento constituem um dos principais assuntos que estudaremos neste capítulo.

[T3] 5.1.4 Implementação do serviço orientado a conexões

No caso do serviço orientado a conexões, precisamos de uma sub-rede de circuitos virtuais. Vejamos como ela funciona. Aidéia que rege os circuitos virtuais é evitar a necessidade de escolher uma nova rota paara cada pacote enviado, como na Figura 5.2. Em vez disso, quando uma conexão é estabelecida, escolhe-se uma rota desde a máquina de origem até a máquina de destino, como

parte da configuração da conexão, e essa rota é armazenada em tabelas internas dos roteadores. A rota é usada por todo o tráfego que flui pela conexão, exatamente como ocorre no sistema telefônico. Quando a conexão é liberada, o circuito virtual também é encerrado. Com o serviço orientado a conexões, cada pacote transporta um identificador, informando a que circuito virtual ele pertence.

Como exemplo, considere a situação da Figura 5.3. Na figura, o host *H*1 estabeleceu a conexão 1 com o host *H*2. Ela é memorizada como a primeira entrada de cada uma das tabelas de roteamento. A primeira linha da tabela de *A* informa que, se um pacote contendo o identificador de conexão 1 chegar de *H*1, ele será enviado ao roteador *C* e receberá o identificador de conexão 1. De modo semelhante, a primeira entrada em *C* faz o roteamento do pacote para *E*, também com o identificador de conexão 1.

Agora, vamos considerar o que acontece se *H*3 também quiser estabelecer uma conexão para *H*2. Ele escolhe o identificador de conexão 1 (porque está iniciando a conexão, e essa é sua única conexão) e informa à sub-rede que ela deve estabelecer o circuito virtual. Isso conduz à segunda linha nas tabelas. Observe que nesse caso temos um conflito porque, embora *A* possa distinguir facilmente os pacotes da conexão 1 provenientes de *H*1 dos pacotes da conexão 1 que vêm de *H*3, *C* não tem como fazer o mesmo. Por essa razão, *A* atribui um identificador de conexão diferente ao tráfego de saída correspondente a segunda conexão. Evitar conflitos desse tipo é a razão pela qual os roteadores precisam ter a capacidade de substituir identificadores de conexões em pacotes de saída. Em alguns contextos, essa operação é chamada troca de rótulos.

[arte: ver original p. 348]

[Dísticos]

[1] Processo P3

[2] Roteador

[3] Equipamento da concessionária de comunicações

[4] Processo P1

[5] Processo P2

[6] Tabela de A

[7] Tabela de C

[8] Tabela de E

[9] Entrada Saída

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.3

[FL] Roteamento em uma sub-rede de circuitos virtuais

[T3] 5.1.5 Comparação entre sub-redes de circuitos virtuais e de datagramas

Tanto os circuitos virtuais quanto os datagramas tem seus simpatizantes e seus detratores. Agora, vamos tentar resumir os argumentos de ambos os lados. As principais questões estão listadas na Figura 5.4, ainda que os puristas provavelmente venham a encontrar um exemplo contrário para tudo que for apresentado na figura.

Dentro da sub-rede, existem vários compromissos entre circuitos virtuais e datagramas como, por exemplo, o compromisso entre espaço de memória do roteador e largura de banda. Os circuitos virtuais permitem que os pacotes contenham números de circuitos em vez de endereços de destino completos. Se os pacotes tenderem a ser muito pequenos, um endereço de destino completo em cada pacote poderá representar um volume significativo de overhead e, portanto, haverá desperdício de largura de banda. O preço pago pelo uso de circuitos

virtuais internamente é o espaço de tabela dentro dos roteadores. Dependendo do custo relativo de circuitos de comunicação em comparação com a memória do roteador, um ou outro pode ser mais econômico.

Outro compromisso é o que se dá entre o tempo de configuração e o tempo de análise de endereço. O uso de circuitos virtuais requer uma fase de configuração, o que leva tempo e consome recursos. Entretanto, é fácil descobrir o que fazer com um pacote de dados em uma sub-rede de circuitos virtuais: o roteador só utiliza o número do circuito para criar um índice em uma tabela e descobrir para onde vai o pacote. Em uma sub-rede de datagramas, é necessário um procedimento de pesquisa mais complicado para localizar a entrada correspondente ao destino.

[arte: ver original p. 349]

[T]Tabela

Questão

Configuração de circuitos

Endereçamento

Informações sobre o estado

Roteamento

Efeito de falhas no roteador

Qualidade de serviço

Controle de congestionamento

Sub-rede de datagramas

Desnecessária

Cada pacote contém os endereços de origem e de destino completos

Os roteadores não armazenam informações sobre o estado das conexões

Cada pacote é roteado independentemente

Nenhum, com exceção dos pacotes perdidos durante a falha

Difícil

Difícil

Sub-rede de circuitos virtuais

Obrigatória

Cada pacote contém um número de circuito virtual curto

Cada circuito virtual requer espaço em tabelas de roteadores por conexão

A rota é escolhida quando o circuito virtual é estabelecido; todos os pacotes seguem essa rota

Todos os circuitos virtuais que tiverem passado pelo roteador que apresentou o defeito serão encerrados

Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual

Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual

[F]Figura 5.4

[FL] Comparação entre sub-redes de circuitos virtuais e de datagramas

Outra questão é a quantidade de espaço de tabelas exigido na memória do roteador. Uma sub-rede de datagramas precisa ter uma entrada para cada destino possível, enquanto uma sub-rede de circuitos virtuais só precisa de uma entrada para cada circuito virtual. Porém, essa vantagem é um tanto ilusória, pois pacotes de configuração de conexões também têm de ser roteados, e eles usam endereços de destino, da mesma forma que os datagramas.

Os circuitos virtuais têm algumas vantagens na garantia de qualidde de serviço e ao evitarem o congestionamento dentro da sub-rede, pois os recursos (por exemplo, buffers, largura de banda e ciclos da CPU) podem ser reservados antecipadamente, quando a conexão é estabelecida. Quando os pacotes

começarem a chegar, a largura de banda e a capacidade do roteador necessárias já estarão instaladas. Com uma sub-rede de datagramas, é mais difícil evitar o congestionamento.

No caso de sistemas de processamento de transações (por exemplo, lojas que telefonam para verificar compras com cartões de crédito), o overhead necessário para configurar e limpar um circuito virtual pode reduzir facilmente o uso do circuito. Caso se espere que a maior parte do tráfego seja desse tipo, o uso de circuitos virtuais comutados dentro da sub-rede fará pouco sentido. Por outro lado, circuitos virtuais permanentes, que são configurados manualmente e duram meses ou anos, talvez sejam úteis nessa situação.

Os circuitos virtuais também têm um problema de vulnerabilidade. Se um roteador apresentar uma falha e perder sua memória, mesmo que volte um segundo depois, todos os circuitos virtuais que estiverem passando por ele terão de ser interrompidos. Por outro lado, se um roteador de datagramas ficar fora do ar, somente os usuários cujos pacotes estiverem enfileirados no roteador naquele momento serão afetados, e talvez nem todos eles, dependendo do fato de já terem sido confirmados ou não. A perda de uma linha de comunicação é fatal para os circuitos virtuais que a utilizam, mas pode ser compensada com facilidade se forem usados datagramas. Os datagramas também permitem que os roteadores equilibrem o tráfego pela sub-rede, pois as rotas podem ser parcialmente alteradas durante uma longa seqüência de transmissões de pacotes.

[T2] 5.2 Algoritmos de roteamento

A principal função da camada de rede é rotear pacotes da máquina de origem para a máquina de destino. Na maioria das sub-redes, os pacotes necessitarão de vários hops para cumprir o trajeto. A única exceção importante diz respeito às redes de difusão, mas mesmo aqui o roteamento depende do fato de a origem e o

destino não estarem na mesma rede. Os algoritmos que escolhem as rotas e as estruturas de dados que eles utilizam constituem um dos elementos mais importantes do projeto da camada de rede.

O **algoritmo de roteamento** é a parte do software da camada de rede responsável pela decisão sobre a linha de saída a ser usada na transmissão do pacote de entrada. Se a sub-rede utilizar datagramas internamente, essa decisão deverá ser tomada mais uma vez para cada pacote de dados recebido, pois a melhor rota pode ter sido alterada desde a última vez. Se a sub-rede utilizar circuitos virtuais internamente, as decisões de roteamento serão tomadas somente quando um novo circuito virtual estiver sendo estabelecido. Daí em diante, os pacotes de dados seguirão a rota previamente estabelecida. Às vezes, essa última circunstância é chamada **roteamento por sessão**, pois uma rota permanece em vigor durante toda uma sessão do usuário (por exemplo, uma sessão de login em um terminal ou uma transferência de arquivos).

Algumas vezes, é útil fazer distinção entre roteamento, que é a tomada de decisão sobre quais rotas utilizar, e encaminhamento, o que acontece quando um pacote chega. Podemos imaginar que um roteador tem dois processos em seu interior. Um deles trata cada pacote que chega, procurando a linha de saída que será usada para ele nas tabelas de roteamento. Esse processo é o **ecaminhamento.** O outro processo é responsável pelo preenchimento e pela atualização das tabelas de roteamento. É nesse processo que o algoritmo de roteamento entra em cena.

Mesmo que as rotas sejam escolhidas independentemente para cada pacote ou apenas quando novas conexões são estabelecidas, certas propriedades são desejáveis em um algoritmo de roteamento: correção, simplicidade, robustez, estabilidade, eqüidade e otimização. Os itens correção e simplicidade são autoexplicativos mas, em princípio, talvez a necessidade de robustez seja menos

óbvia. Uma vez que uma rede de maior porte é instalada, espera-se que ela funcione continuamente durante anos sem apresentar qualquer falha no sistema. Durante esse período, haverá falhas de hardware e software de todos os tipos. Os hosts, os roteadores e as linhas irão falhar repetidamente, e a topologia mudará muitas vezes. O algoritmo de roteamento deve ser capaz de aceitar as alterações na topologia e no tráfego sem exigir que todas as tarefas de todos os hosts sejam interrompidas e que a rede seja reinicializada sempre que algum roteador apresentar falha.

A estabilidade também é um objetivo importante do algoritmo de roteamento. Existem algoritmos que nunca convergem para o equilíbrio, independente do tempo em que são executados. Um algoritmo estável alcança o equilíbrio e permanece nesse estado. A eqüidade e a otimização podem parecer óbvias, pois certamente ninguém faria oposição a elas. No entanto, como se vê, com freqüência elas têm objetivos contraditórios. Como um exemplo simples desse conflito, observe a Figura 5.5. Suponha que o tráfego entre *A* e *A'*, entre *B* e *B'* e entre *C* e *C'* seja suficiente para saturar os enlaces horizontais. Para maximizar o fluxo total, o tráfego de *X* para *X'* deve ser desativado por completo. Infelizmente, talvez *X* e *X'* não vejam a situação dessa maneira. É evidente que se faz necessário um meio-termo entre eficiência global e eqüidade para as conexões individuais.

[arte: ver original p. 351]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.5

[FL] Conflito entre eqüidade e otimização

Antes de tentarmos encontrar um meio-termo entre eqüidade e otimização, devemos decidir o que estamos buscando otimizar. A minimização do retardo médio de pacote é uma candidata óbvia, e o mesmo vale para a maximização do throughput total da rede. Além disso, esse dois objetivos também estão em conflito, pois operar qualquer sistema de enfileiramento em uma velocidade próxima a de sua capacidade máxima implica um longo retardo de enfileiramento. Como meio-termo, muitas redes tentam minimizar o número de hops que um pacote deve percorrer, pois a redução do número de hops tende a melhorar o retardo e também a reduzir o volume de largura de banda consumido o que, por sua vez, também tende a melhorar o throughput.

Os algoritmos de roteamento podem ser agrupados em duas classes principais: adaptativos e não adaptativos. Os **algoritmos não adaptativos** não baseiam suas decisões de roteamento em medidas ou estimativas do tráfego e da topologia atuais. Em vez disso, a escolha da rota a ser utilizada para ir de *I* até *J* (para todo *I* e todo *J*) é previamente calculada off-line, sendo transferida para os roteadores quando a rede é inicializada. Às vezes, esse procedimento é chamado **roteamento estático**.

Em contraste, os algoritmos adaptativos mudam suas decisões de roteamento para refletir mudanças na topologia e, normalmente, também no tráfego. Os algoritmos adaptativos diferem em termos do lugar em que obtêm suas informações (por exemplo, no local, de roteadores adjacentes ou de todos os roteadores), do momento em que alteram as rotas (por exemplo, a cada ΔT segundos, quando a carga se altera ou quando a topologia muda) e da unidade métrica utilizada para a otimização (por exemplo, distância, número de hops ou tempo de trânsito estimado). Nas próximas seções, trataremos de uma variedade de algoritmos de roteamento, tanto estáticos quanto dinâmicos.

[T3] 5.2.1 O princípio de otimização

Antes de estudarmos algoritmos específicos, talvez valha a pena lembrar que é possível criar uma descrição geral das rotas ótimas sem levar em conta a topologia ou o tráfego de rede. Essa descrição é conhecida como **princípio de otimização.** Esse princípio estabelece que, se o roteador *J* estiver no caminho ótimo entre o roteador *I* e o roteador *K*, o caminho ótimo de *J* até *K* também estará na mesma rota. Para confirmar isso, chame a parte da rota entre *I* e *J* de r_1 e o restante de r_2. Se existisse uma rota melhor que r_2 entre *J* e *K*, ela poderia ser concatenada com r_1 para melhorar a rota entre *I* e *K*, contradizendo nossa afirmação de que $r_1 r_2$ é ótima.

Como conseqüência direta do princípio de otimização, podemos observar que o conjunto de rotas ótimas de todas as origens para um determinado destino forma uma árvore com raiz no destino. Uma árvore como essa é chamada **árvore de escoamento** e está ilustrada na Figura 5.6, onde a unidade métrica de distância é o número de hops. Observe que uma árvore de escoamento não é necessariamente exclusiva; podem existir outras árvores com caminhos de mesmo tamanho. O objetivo de todos os algoritmos de roteamento é descobrir e utilizar as árvores de escoamento em todos os roteadores.

[arte: ver original p. 352]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.6

[FL] (a) Uma sub-rede. (b) Uma árvore de escoamento para o roteador *B*

Como uma árvore de escoamento é de fato uma árvore, ela não contém loops; portanto, cada pacote será entregue dentro de um número finito e limitado de

hops. Na prática, nem tudo é tão fácil assim. Enlaces e roteadores podem sair do ar e voltar à atividade durante a operação; desse modo, diferentes roteadores podem ter idéias distintas sobre a topologia atual. Além disso, empregamos alguns artifícios para resolver a seguinte questão: cada roteador deve obter individualmente as informações sobre a base do cálculo de sua árvore de escoamento, ou esses dados serão obtidos por algum outro meio. Retornaremos a essas questões em breve. Contudo, o princípio da otimização e a árvore de escoamento permitem que se faça um teste de benchmark para detectar que outros algoritmos de roteamento podem ser medidos.

[T3] 5.2.2 Roteamento pelo caminho mais curto

Vamos iniciar nosso estudo de algoritmos de roteamento práticos com uma técnica amplamente utilizada em muitas formas, pois é simples e fácil de entender. A idéia é criar um grafo da sub-rede, com cada nó do grafo representando um roteador e cada arco indicando uma linha de comunicação (geralmente chamada enlace). Para escolher uma rota entre um determinado par de roteadores, o algoritmo simplesmente encontra o caminho mais curto entre eles no grafo.

O conceito de **caminho mais curto** merece uma explicação. Uma forma de medir o comprimento do caminho é usar o número de hops. Empregando-se essa unidade de medida, os caminhos *ABC* e *ABE* da Figura 5.7 são igualmente longos. Uma outra unidade métrica é a distância geográfica em quilômetros, e nesse caso *ABC* é claramente muito mais longo que *ABE* (supondo-se que a figura tenha sido desenhada em escala).

Entretanto, muitas outras unidades métricas também são possíveis além do número de hops e da distância física. Por exemplo, cada arco poderia ser identificado com o retardo médio de enfileiramento e de transmissão referente a

um pacote de teste padrão, de acordo com as especificações de testes executados a cada hora. Nesse grafo, o caminho mais curto é o caminho mais rápido, e não o caminho com menor número de arcos ou quilômetros.

No caso geral, os rótulos dos arcos podem ser calculados como uma função da distância, da largura de banda, do tráfego médio, do custo de comunicação, do comprimento médio da fila, do retardo medido e de outros fatores. Alterando-se a função de ponderação (atribuição de pesos), o algoritmo calcularia o caminho "mais curto" medido de acordo com qualquer critério ou com uma combinação de critérios.

São conhecidos diversos algoritmos para se calcular o caminho mais curto entre dois nós de um grafo. O algoritmo que vamos examinar agora se deve a Dijkstra (1959). Cada nó é identificado (entre parênteses) por sua distância a partir do nó de origem ao longo do melhor caminho conhecido. Inicialmente, nenhum caminho é conhecido; portanto, todos os nós são rotulados com infinito. À medida que o algoritmo prossegue e os caminhos são encontrados, os rótulos podem mudar, refletindo melhores caminhos. Um rótulo pode ser provisório ou permanente. No início, todos são provisórios. Quando se descobre que um rótulo representa o caminho mais curto possível até a origem desse nó, ele se torna permanente e nunca mais é alterado daí em diante.

Para ilustrar como funciona o algoritmo de identificação, vamos examinar o grafo ponderado não orientado mostrado na Figura 5.7(a), onde os pesos representam, por exemplo, a distância. Desejamos encontrar o caminho mais curto de *A* até *D*. Começamos marcando o nó *A* como permanente, o que é indicado por um círculo preenchido. Depois examinamos separadamente cada um dos nós adjacentes a *A* (o nó ativo), alterando o rótulo de cada um deles para indicar a distância até *A*. Sempre que um nó é rotulado novamente, ele também é rotulado com o nó a partir do qual o teste foi feito; assim, podemos reconstruir o caminho final mais

tarde. Após examinarmos cada um dos nós adjacentes a *A*, verificamos todos os nós provisoriamente rotulados no grafo inteiro e tornamos permanente o nó que tem o menor rótulo, como mostra a Figura 5.7(b). Esse nó passa a ser o novo nó ativo.

[arte: ver original p. 354]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.7

[FL] As cinco primeiras etapas utilizadas no cálculo do caminho mais curto de *A* até *D*. As setas indicam o nó ativo

Agora, começamos por *B* e examinamos todos os nós adjacentes a ele. Se a soma do rótulo de *B* e a distância entre *B* e o nó que está sendo considerado for menor que o rótulo desse nó, teremos um caminho mais curto; portanto, o nó será rotulado novamente.

Depois que todos os nós adjacentes ao nó ativo tiverem sido inspecionados e os rótulos provisórios tiverem sido alterados na medida do possível, o grafo inteiro será pesquisado até ser encontrado o nó com o rótulo provisório de menor valor. Esse nó passará a ser o nó permanente e se tornará o nó ativo na próxima iteração. A Figura 5.7 mostra as cinco primeiras etapas do algoritmo.

Para saber por que o algoritmo funciona, observe a Figura 5.7(c). Nesse momento, tornamos *E* permanente. Suponha a existência de um caminho mais curto que *ABE*, digamos, *AXYZE*. Há duas possibilidades: o nó *Z* já se tornou permanente, ou o nó *Z* não se tornou permanente. Se ele já se tornou permanente, então *E* já foi testado (na iteração que se segue àquela em que *Z* se tornou permanente); assim, o caminho *AXYZE* não escapou à nossa atenção, e

portanto não pode ser um caminho mais curto.

Agora, leve em conta a hipótese de *Z* ainda ter um rótulo provisório. Então, o rótulo *Z* é maior ou igual ao de *E* e, nesse caso, *AXYZE* não pode ser um caminho mais curto que *ABE*, ou ele é menor que o de *E* e, nesse caso, *Z* e não *E* se tornará permanente primeiro, permitindo que *E* seja testado a partir de *Z*.

Esse algoritmo é mostrado na Figura 5.8. As variáveis globais *n* e *dist* descrevem o grafo e são inicializadas antes de *shortest_path* ser chamado. A única diferença entre o programa e o algoritmo descrito anteriormente é que, na Figura 5.8, calculamos o caminho mais curto a partir do nó terminal *t*, em vez de começarmos no nó de origem, *s*. Como o caminho mais curto de *t* até *s* em um grafo não orientado é igual ao caminho mais curto de *s* até *t*, não importa em que extremidade começamos (a menos que haja diversos caminhos mais curtos e, nesse caso, a inversão da pesquisa possa descobrir um caminho diferente). A razão para a pesquisa no sentido inverso é que cada nó é rotulado com seu predecessor em vez de ser rotulado com seu sucessor. Quando o caminho final for copiado na variável de saída *path*, o caminho será então invertido. Invertendo-se o sentido da pesquisa, os dois efeitos se cancelarão, e a resposta será produzida na ordem correta.

[T3] 5.2.3 Inundação

Outro algoritmo estático é o algoritmo de **inundação (flooding)**, no qual cada pacote de entrada é enviado para toda linha de saída, exceto para aquela em que chegou. Evidentemente, o algoritmo de inundação gera uma vasta quantidade de pacotes duplicados, na verdade um número infinito, a menos que algumas medidas sejam tomadas para tornar mais lento o processo. Uma dessas medidas é ter um contador de hops contido no cabeçalho de cada pacote; o contador é decrementado em cada hop, com o pacote sendo descartado quando o contador

atingir zero. O ideal é que o contador de hops seja inicializado com o comprimento do caminho desde a origem até o destino. Se não souber o tamanho do caminho, o transmissor poderá inicializar o contador com o valor referente ao pior caso, ou seja, o diâmetro total da sub-rede.

Uma técnica alternativa para conter o processo de inundação é controlar quais pacotes foram transmitidos por inundação, a fim de evitar transmiti-los uma segunda vez. Uma forma de conseguir isso é fazer o roteador de origem inserir um número de seqüência em cada pacote recebido de seus hosts. Portanto, cada roteador precisará de uma lista por roteador de origem informando quais números de seqüência originários desse ponto já foram vistos. Se houver um pacote de entrada na lista, ele não será transmitido na inundação.

Para evitar que as listas cresçam indefinidamente, cada lista deve ser incrementada de acordo com um contador *k*, o que significa que todos os números de seqüência até *k* foram vistos. Quando um pacote for recebido, será fácil verificar se ele é uma cópia; se for, ele será descartado. Além disso, a lista completa abaixo de *k* não é necessária, visto que *k* na verdade resume essa lista.

Uma variação um pouco mais prática do algoritmo de inundação é a **inundação seletiva.** Nesse algoritmo, os roteadores não enviam cada pacote de entrada para todas as linhas, apenas para aquelas que provavelmente estão na direção certa. Em geral, não há muita razão para se utilizar uma linha da região leste para transportar um pacote cujo destino seja a região oeste, a menos que a topologia seja extremamente peculiar e o roteador esteja seguro desse fato.

[arte: ver original da p. 356]

[TD]

```
#define MAX_NODES 1024                /* número máximo  de nós */
#define INFINITY 1000000000           /* um número maior que todos os
caminhos máximos */
```

```
int n, dist[MAX_NODES][MAX_NODES];          /* dist[i][j] é a distância de i até j */

void shortest_path(int s, int t, int path[])

{ struct state {                              /* o caminho em que estamos
trabalhando */
      int predecessor;                        /* nó anterior */
      int length;                             /* distância desde a origem até este
nó */
      enum {permanent, tentative} label;      /* estado do rótulo (label) */
  } state[MAX_NODES];
  int i, k, min;
  struct state *p;
  for (p = &state[0]; p < &state[n]; p++) {   /* inicializa estado */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                      /* k é o nó  ativo inicial */
  do {                                        /* Existe um caminho melhor a
partir de k? */
      for (i = 0; i < n; i++)                 /* esse grafo tem n nós */
           if (dist[k][i] != 0 && state[i].label == tentative) {
                if (state[k].length + dist[k][i] < state[i].length) {
                     state[i].predecessor = k;
                     state[i].length = state[k].length + dist[k][i];
                }
           }
```

```
      /* Encontra o nó rotulado provisoriamente com o menor rótulo. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
            if (state[i].label == tentative && state[i].length < min) {
                  min = state[i].length;
                  k = i;
            }
      state[k].label = permanent;
  } while (k != s);

  /* Copia o caminho no array de saída. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
} [TN]
```

[F]Figura 5.8

[FL] O algoritmo de Dijkstra para calcular o caminho mais curto através de um grafo

O algoritmo de inundação não é prático na maioria das aplicações, mas tem sua utilidade. Por exemplo, em aplicações militares, em que muitos roteadores podem ser destruídos a qualquer momento, a grande robustez do algoritmo de inundação é altamente desejável. Em aplicações de bancos de dados distribuídos, às vezes é necessário atualizar todos os bancos de dados ao mesmo tempo e, nesse caso, o algoritmo de inundação pode ser bastante útil. em redes sem fios, todas as mensagens transmitidas por uma estação podem ser recebidas por todas as outras estações dentro de seu alcance de rádio; na verdade, isso representa a

inundação, e alguns algoritmos empregam essa propriedade. Um quarto uso possível do algoritmo de inundação é como uma unidade de medida que servirá como base de comparação com outros algoritmos de roteamento. O algoritmo de inundação sempre escolhe o caminho mais curto, pois todos os caminhos possíveis são selecionados em paralelo. Em conseqüência disso, nenhum outro algoritmo é capaz de produzir um retardo de menor duração (se ignorarmos o overhead gerado pelo próprio processo de inundação).

[T3] 5.2.4 Roteamento com vetor de distância

Geralmente, as modernas redes de computadores utilizam algoritmos de roteamento dinâmicos em lugar dos algoritmos estáticos descritos aqui, porque os algoritmos estáticos não levam em conta a carga atual da rede. Dois algoritmos dinâmicos específicos, o roteamento com vetor de distância e o roteamento por estado de enlace, são os mais conhecidos. Nesta seção, vamos estudar o primeiro desses algoritmos. Na próxima seção, estudaremos o segundo.

Os algoritmos de **roteamento com vetor de distância** operam fazendo cada roteador manter uma tabela (isto é, um vetor) que fornece a melhor distância conhecida até cada destino e determina qual linha deve ser utilizada para se chegar lá. Essas tabelas são atualizadas através da troca de informações com os vizinhos.

Às vezes, o algoritmo de roteamento com vetor de distância recebe outros nomes, sendo mais comuns o algoritmo de roteamento distribuído de **Bellman-Ford** e o algoritmo de **Ford-Fulkerson**. Esses algoritmos receberam os nomes dos pesquisadores que os desenvolveram (Bellman, 1957; e Ford e Fulkerson, 1962).

O algoritmo de roteamento com vetor de distância era o algoritmo de roteamento original da ARPANET, e também foi utilizado na Internet, com o nome RIP.

No roteamento com vetor de distância, cada roteador mantém uma tabela de roteamento indexada por cada roteador da sub-rede e que contém uma entrada para cada um desses roteadores. Essa entrada contém duas partes: a linha de saída preferencial a ser utilizada para esse destino e uma estimativa do tempo ou da distância até o destino. A unidade métrica utilizada pode ser o número de hops, o retardo de tempo em milissegundos, o número total de pacotes enfileirados no caminho ou algo semelhante.

Presume-se que o roteador conheça a "distância" até cada um de seus vizinhos. Se a unidade métrica for o hop, a distância será de apenas um hop. Se a unidade métrica for o comprimento da fila, o roteador simplesmente examinará cada uma das filas. Se a unidade métrica for o retardo, o roteador poderá medi-lo diretamente com pacotes ECHO que o receptor irá identificar com um timbre de hora e transmitir de volta o mais rápido que puder.

Por exemplo, suponha que o retardo seja usado como unidade métrica e que o roteador saiba qual é o retardo até cada um de seus vizinhos. Uma vez a cada T ms, cada roteador envia a cada vizinho uma lista de seus retardos estimados até cada destino. O roteador também recebe uma lista semelhante de cada vizinho. Imagine que uma dessas tabelas tenha acabado de chegar do vizinho X, sendo X_i a estimativa de X sobre o tempo que ele levará para chegar até o roteador i. Se o roteador souber que o retardo para X é de m ms, ele também saberá que pode alcançar o roteador i por meio de X em $X_i + m$ ms. Efetuando esse cálculo para cada vizinho, um roteador pode descobrir qual estimativa parece ser a melhor, e também pode usar essa estimativa e a linha correspondente em sua nova tabela de roteamento. Observe que a antiga tabela de roteamento não é utilizada no cálculo.

Esse processo de atualização é ilustrado na Figura 5.9. A parte (a) mostra uma sub-rede. As quatro primeiras colunas da parte (b) mostram os vetores de retardo

recebidos dos vizinhos do roteador *J*. *A* alega ter um retardo de 12 ms até *B*, um retardo de 25 ms até *C*, um retardo de 40 ms até *D* etc. Suponha que *J* tenha medido ou estimado seu retardo até seus vizinhos *A*, *I*, *H* e *K* como 8, 10, 12 e 6 ms, respectivamente.

[arte: ver original p. 358]

[Dísticos]

[1] Roteador

[2] Para

[3] Novo retardo estimado a partir de J

[4] Linha

[5] O retardo de JA é 8

[6] O retardo de JI é 10

[7] O retardo de JH é 12

[8] O retardo de JK é 6

[9] Vetores recebidos de quatro vizinhos de J

[10] Nova tabela de roteamento para J

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.9

[FL] (a) Uma sub-rede. (b) Entrada de *A*, *I*, *H*, *K* e a nova tabela de roteamento para *J*

Considere a forma como *J* calcula sua nova rota até o roteador *G*. Ele sabe que pode chegar até *A* em 8 ms e *A* alega ser capaz de chegar a *G* em 18 ms; portanto, *J* sabe que pode contar com um retardo de 26 ms até *G*, se encaminhar pacotes destinados a *G* para *A*. Da mesma forma, ele calcula o retardo para *G* via

I, *H* e *K* como 41 (31 + 10), 18 (6 + 12) e 37 (31 + 6) ms, respectivamente. O melhor desses valores é 18; portanto, *J* cria uma entrada em sua tabela de roteamento indicando que o retardo até *G* é 18 ms e que a rota a ser utilizada passa por *H*. O mesmo cálculo é feito para todos os outros destinos, sendo a nova tabela de roteamento mostrada na última coluna da figura.

[T4] O problema da contagem até infinito

O roteamento com vetor de distância funciona na teoria, mas tem um sério inconveniente na prática: apesar de convergir para a resposta correta, ele pode fazê-lo muito lentamente. Em particular, ele reage com rapidez a boas notícias, mas reage devagar a más notícias. Imagine um roteador cuja melhor rota até o destino *X* seja grande. Se na próxima troca, o vizinho *A* repentinamente relatar um pequeno retardo até *X*, o roteador deixará de usar a linha que vai até *A* e enviará o tráfego para *X*. Em uma troca de vetores, a boa notícia é sempre processada.

Para ver a que velocidade as boas notícias se propagam, considere a sub-rede de cinco nós (linear) da Figura 5.10, na qual a unidade métrica para calcular o retardo é o número de hops. Suponha que *A* inicialmente esteja inativo e que todos os outros roteadores saibam disso. Em outras palavras, todos eles registraram o retardo até *A* como infinito.

[arte: ver original p. 359]

[Dísticos]

[1]

A	B	C	D	E	
	•	•	•	•	Inicialmente
	1	•	•	•	Após 1 troca
	1	2	•	•	Após 2 trocas

1	2	3	•	Após 3 trocas
1	2	3	4	Após 4 trocas

(a)

[2]

A	B	C	D	E	
	1	2	3	4	Inicialmente
	3	2	3	4	Após 1 troca
	3	4	3	4	Após 2 trocas
	5	4	5	4	Após 3 trocas
	5	6	5	6	Após 4 trocas
	7	6	7	6	Após 5 trocas
	7	8	7	8	Após 6 trocas
		.			
		.			
		.			
	•	•	•	•	

(b)

[F]Figura 5.10

[FL] O problema da contagem até infinito

Quando *A* está ativo, os outros roteadores tomam conhecimento dele através de trocas de vetores. Para simplificar, vamos supor que exista um gongo gigantesco em algum lugar e que ele seja tocado periodicamente para dar início a uma troca de vetores em todos os roteadores ao mesmo tempo. No momento da primeira troca, *B* toma conhecimento de que seu vizinho da esquerda tem retardo zero até *A*. Agora, *B* cria uma entrada em sua tabela de roteamento, indicando que *A* está a um hop de distância à esquerda. Todos os outros roteadores continuam

imaginando que *A* está inativo. Nesse momento, as entradas da tabela de roteamento correspondentes a *A* são as que estão ilustradas na segunda linha da Figura 5.10(a). Na troca seguinte, *C* toma conhecimento de que *B* tem um caminho de comprimento 1 até *A*; portanto, *C* atualiza sua tabela de roteamento para indicar um caminho de comprimento 2, mas *D* e *E* só detectam as boas notícias mais tarde. É claro que as boas notícias estão sendo espalhadas na velocidade de um hop por troca. Em uma sub-rede cujo caminho mais longo tem o comprimento de *N* hops, dentro de *N* trocas, todos saberão quais linhas e roteadores foram recentemente reativados.

Agora, vamos considerar a situação da Figura 5.10(b), em que todas as linhas e roteadores estão inicialmente ativos. Os roteadores *B*, *C*, *D* e *E* têm distâncias até *A* iguais a 1, 2, 3 e 4, respectivamente. De repente *A* é desativado, ou então a linha entre *A* e *B* é interrompida, o que efetivamente é o mesmo, do ponto de vista de *B*.

Na primeira troca de pacotes, *B* nada detecta em *A*. Felizmente, *C* informa: Não se preocupe. Tenho um caminho até *A* de comprimento 2. De pouco adianta *B* saber que o caminho de *C* passa pelo próprio *B*. Apesar de *B* saber disso, *C* pode ter 10 linhas de saída, todas com caminhos independentes até *A* de comprimento 2. Em conseqüência disso, *B* agora imagina que pode atingir *A* via *C*, com um comprimento de caminho igual a 3. *D* e *E* não atualizam suas entradas correspondentes a *A* na primeira troca.

Na segunda troca, *C* percebe que cada um de seus vizinhos alega ter um caminho até *A* de comprimento 3. *C* seleciona um desses caminhos ao acaso e torna 4 sua nova distância até *A*, como mostra a terceira fileira da Figura 5.10(b). As trocas subseqüentes produzem o histórico mostrado no restante da Figura 5.10(b).

A partir dessa figura, deve ficar claro por que as más notícias se propagam lentamente: nenhum roteador tem um valor maior que uma unidade a mais que o

valor mínimo de todos os seus vizinhos. Gradualmente, todos os roteadores seguem seu caminho até infinito, mas o número de trocas necessárias depende do valor numérico utilizado para infinito. Por essa razão, é melhor definir infinito como o caminho mais longo mais 1 unidade. Se a unidade métrica for o retardo de tempo, não haverá um limite superior bem definido; assim, será necessário um valor alto para evitar que um caminho com um retardo longo seja tratado como inativo. Não é de surpreender totalmente que esse problema seja conhecido como problema da **contagem até infinito**. Houve algumas tentativas de resolvê-lo (como a de divisão horizontal de inversão envenenada, descrita na RFC 1058), mas nenhuma delas funcionou bem no caso geral. O núcleo do problema é que, quando *X* informa a *Y* que tem um caminho em algum lugar, *Y* não tem como saber se ele próprio está no caminho.

[T3] 5.2.5 Roteamento por estado de enlace

O roteamento com vetor de distância foi utilizado na ARPANET até 1979, quando foi substituído pelo roteamento por estado de enlace. Essa substituição foi motivada por dois problemas principais. Primeiro, como a unidade métrica de retardo era o comprimento da fila, não se levava em conta a largura de banda da linha quando se escolhiam rotas. Como no início todas as linhas tinham 56 kbps, a largura de banda das linhas não era importante mas, após a atualização de algumas linhas para 230 kbps e de outras para 1,544 Mbps, não considerar a largura de banda se tornou um problema importante. É claro que teria sido possível mudar a unidade métrica do retardo para decompor em fatores a largura de banda. No entanto, havia um segundo problema, ou seja, o algoritmo geralmente levava muito tempo para convergir (o problema da contagem até infinito). Por essas razões, ele foi substituído por um algoritmo inteiramente novo, agora chamado **roteamento por estado de enlace**. Variantes do roteamento

por estado de enlace agora são amplamente utilizadas.

A idéia por trás do roteamento por estado de enlace é simples e pode ser estabelecida como cinco partes. Cada roteador deve fazer o seguinte:

1. Descobrir seus vizinhos e aprender seus endereços de rede.
2. Medir o retardo ou o custo até cada um de seus vizinhos.
3. Criar um pacote que informe tudo o que ele acabou de aprender.
4. Enviar esse pacote a todos os outros roteadores.
5. Calcular o caminho mais curto até cada um dos outros roteadores.

Com efeito, a topologia completa e todos os retardos são medidos experimentalmente e distribuídos para todos os outros roteadores. Em seguida, o algoritmo de Dijkstra pode ser usado para encontrar o caminho mais curto até cada um dos outros roteadores. Em seguida, estudaremos cada uma dessas cinco etapas em detalhes.

[T4] Conhecendo os vizinhos

Quando um roteador é inicializado, sua primeira tarefa é aprender quem são seus vizinhos. Esse objetivo é alcançado enviando-se um pacote HELLO especial em cada linha ponto a ponto. O roteador da outra extremidade deve enviar de volta uma resposta, informando quem é. Esses nomes devem ser globalmente exclusivos pois, quando um roteador distante ouvir mais tarde que esses três roteadores estão todos conectados a *F*, é essencial que ele possa determinar se todos os três representam o mesmo *F*.

Quando dois ou mais roteadores estão conectados por uma LAN, a situação é um pouco mais complicada. A Figura 5.11(a) ilustra uma LAN à qual três roteadores, *A*, *C* e *F*, estão diretamente conectados. Cada um desses roteadores está conectado a um ou mais roteadores adicionais, como é mostra a figura.

[arte: ver original p. 361]

[Dísticos]

[1] Roteador

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.11

[FL] (a) Nove roteadores e uma LAN. (b) Um modelo de grafo de (a)

Uma forma de modelar a LAN é considerá-la como um nó, conforme mostra a Figura 5.11(b). Aqui, introduzimos um novo nó artificial *N*, ao qual *A*, *C* e *F* estão conectados. A possibilidade de ir de *A* até *C* na LAN é representada aqui pelo caminho *ANC*.

[T4] Como medir o custo da linha

O algoritmo de roteamento por estado de enlace exige que cada roteador conheça o retardo para cada um de seus vizinhos (ou pelo menos tenha uma boa estimativa de qual seja ele). A forma mais simples de determinar esse retardo é enviar um pacote especial ECHO pela linha que o outro lado deve transmitir de volta imediatamente. Medindo o tempo de ida e volta e dividindo-o por dois, o roteador transmissor pode obter uma estimativa razoável do retardo. Para obter resultados ainda melhores, é possível realizar o teste várias vezes e usar a média obtida. É claro que esse método pressupõe implicitamente que os retardos são simétricos, o que nem sempre ocorre.

Uma questão interessante é se a carga deve ou não ser levada em consideração ao ser medido o retardo. Para decompor a carga fatores, o timer encarregado de medir o tempo de ida e volta deve ser iniciado quando o pacote ECHO for enfileirado. Para ignorar a carga, o timer deve ser iniciado quando o pacote ECHO

atingir o início da fila.

Há argumentos a favor das duas opções. A inclusão dos retardos induzidos pelo tráfego nas medições significa que, quando um roteador tiver de escolher entre duas linhas com a mesma largura de banda, uma das quais pesadamente carregada durante todo o tempo, ao contrário da outra, o roteador irá considerar a rota sobre a linha não carregada um caminho mais curto. Essa opção resultará em melhor desempenho.

Infelizmente, também há um argumento contra a inclusão da carga no cálculo do retardo. Tome como exemplo a sub-rede da Figura 5.12, que é dividida em duas partes, Leste e Oeste, interconectadas por duas linhas, *CF* e *EI*.

[arte: ver original p. 362]

[Dísticos]

[1] Oeste

[2] Leste

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.12

[FL] Uma sub-rede em que as partes Leste e Oeste estão conectadas por duas linhas

Suponha que a maior parte do tráfego entre as regiões Leste e Oeste esteja usando a linha *CF* e que, como resultado, essa linha esteja fortemente carregada com longos retardos. A inclusão do retardo de enfileiramento no cálculo do caminho mais curto tornará *EI* mais atraente. Depois de instaladas as novas tabelas de roteamento, a maior parte do tráfego Leste-Oeste será conduzida através de *EI*, sobrecarregando essa linha. Conseqüentemente, na próxima

atualização, *CF* parecerá ser o caminho mais curto. Como resultado, as tabelas de roteamento poderão oscilar muito, causando um roteamento errático e muitos problemas potenciais. Se a carga for ignorada e considerarmos apenas a largura de banda, esse problema não ocorrerá. Como alternativa, a carga poderá ser distribuída entre as duas linhas, mas essa solução não utiliza totalmente o melhor caminho. Apesar disso, para evitar oscilações na escolha do melhor caminho, talvez seja sensato distribuir a carga por várias linhas, com alguma fração conhecida dessa carga sendo transmitida através de cada linha.

[T4] Como criar pacotes de estado de enlace

Uma vez obtidas as informações necessárias para a troca, a próxima etapa é cada roteador criar um pacote que contenha todos os dados. O pacote começa com a identidade do transmissor, seguida por um número de seqüência, e pela idade (a ser descrita mais adiante) e por uma lista de vizinhos. É fornecido o retardo referente a cada vizinho. Um exemplo de sub-rede é apresentado na Figura 5.13(a), sendo os retardos mostrados como rótulos nas linhas. Os pacotes de estado de enlace correspondentes a todos os seis roteadores são mostrados na Figura 5.13(b).

[arte: ver original p. 363]

[Dísticos]

[1] B 2 C

4 3

A D

5 1 6 7

E 8 F

(a)

[2]

Enlace Estado Pacotes

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Idade	Idade	Idade	Idade	Idade	Idade
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

(b)

[F]Figura 5.13

[FL] (a) Uma sub-rede. (b) Os pacotes de estado de enlace correspondentes a essa sub-rede

É fácil criar os pacotes de estado de enlace. Difícil é determinar quando criá-los. Uma possibilidade é criá-los periodicamente, ou seja, a intervalos regulares. Outra possibilidade é criá-los durante a ocorrência de algum evento significativo, como uma linha ou vizinho que sai do ar, entra em atividade novamente ou altera suas propriedades de forma apreciável.

[T4] Distribuição dos pacotes de estado de enlace

A parte mais complicada do algoritmo é distribuir os pacotes de estado de enlace de forma confiável. À medida que os pacotes são distribuídos e instalados, os roteadores que obtiverem os primeiros pacotes mudarão suas rotas. Conseqüentemente, os diferentes roteadores talvez estejam usando diferentes versões da topologia, o que poderá levar a inconsistências, loops, máquinas inacessíveis e outros problemas.

Primeiro, descreveremos o algoritmo básico de distribuição. Depois, vamos aperfeiçoá-lo. A idéia fundamental é usar o algoritmo de inundação para

distribuir os pacotes de estado de enlace. Para manter o controle do processo do algoritmo de inundação, cada pacote contém um número de seqüência que é incrementado para cada novo pacote enviado. Os roteadores controlam todos os pares (roteador de origem, seqüência) que vêem. Quando é recebido, o novo pacote de estado de enlace é conferido na lista de pacotes já verificados. Se for novo, ele será encaminhado a todas as linhas, exceto à linha por onde chegou. Se for uma cópia, o pacote será descartado. Se um pacote recebido tiver com número de seqüência mais baixo que o mais alto número de seqüência detectado até o momento, ele será rejeitado e considerado obsoleto, pois o roteador terá dados mais recentes.

Esse algoritmo apresenta alguns problemas, mas eles são contornáveis. Primeiro, se os números de seqüência se repetirem, a confusão imperará. A solução aqui é usar um número de seqüência de 32 bits. Com um pacote de estado de enlace por segundo, seriam necessários 137 anos para um número se repetir; portanto, essa possibilidade pode ser ignorada.

Em segundo lugar, se um roteador apresentar falha, ele perderá o controle de seu número de seqüência. Se ele começar de novo em 0, o pacote seguinte será rejeitado por ser considerado uma cópia.

Em terceiro lugar, se um número de seqüência for adulterado e o número 65.540 for recebido no lugar do número 4 (um erro de 1 bit), os pacotes de 5 a 65.540 serão rejeitados como obsoletos, pois 65.540 será considerado o número de seqüência atual.

A solução para todos esses problemas é incluir a idade de cada pacote após o número de seqüência e decrementá-la uma vez por segundo. Quando a idade atingir zero, as informações desse roteador serão descartadas. Normalmente, um novo pacote chega, digamos, a cada 10 segundos; logo, as informações do roteador só alcançarão o timeout (tempo limite) quando um roteador estiver

inativo (ou quando seis pacotes consecutivos se perderem, um evento improvável). O campo *Idade* também é decrementado por cada roteador durante o processo inicial de inundação para garantir que nenhum pacote será perdido e irá durar um período de tempo indefinido (um pacote cuja idade for zero será descartado).

Alguns aprimoramentos nesse algoritmo o tornam mais resistente. Quando um pacote de estado de enlace chega a um roteador para inundação, ele não é imediatamente enfileirado para transmissão. Em vez disso, ele é colocado em uma área de retenção para aguardar um pouco. Se outro pacote de estado de enlace da mesma origem chegar antes da transmissão do primeiro pacote, seus números de seqüência serão comparados. Se forem iguais, a cópia será descartada. Se forem diferentes, o mais antigo será descartado. Para evitar erros nas linhas entre dois roteadores, todos os pacotes de estado de enlace são confirmados. Quando uma linha ficar ociosa, a área de retenção será varrida seqüencialmente, a fim de se selecionar um pacote ou uma confirmação a enviar.

A estrutura de dados utilizada pelo roteador *B* da sub-rede mostrada na Figura 5.13(a) é representada na Figura 5.14. Cada linha aqui corresponde a um pacote de estado de enlace recém-chegado, mas ainda não totalmente processado. A tabela registra a origem do pacote, seu número de seqüência e idade, e os dados correspondentes. Além disso, há flags de transmissão e confirmação para cada uma das três linhas de *B* (para *A*, *C* e *F*, respectivamente). Os flags de transmissão significam que o pacote deve ser enviado na linha indicada. Os flags de confirmação significam que ele deve ser confirmado ali.

Na Figura 5.14, o pacote de estado de enlace de *A* chega diretamente; portanto, ele deve ser enviado para *C* e *F* e confirmado para *A*, como indicam os bits dos flags. Da mesma forma, o pacote proveniente de *F* deve ser encaminhado para *A* e *C*, e confirmado para *F*.

Entretanto, a situação com o terceiro pacote, proveniente de *E*, é diferente. Ele chegou duas vezes, uma vez através de *EAB* e outra através de *EFB*. Conseqüentemente, ele só precisa ser enviado para *C*, mas deve ser confirmado para *A* e *F*, conforme indicam os bits.

Se uma cópia for recebida enquanto o original ainda estiver no buffer, os bits deverão ser alterados. Por exemplo, se uma cópia do estado de *C* chegar de *F* antes da quarta entrada da tabela ter sido encaminhada, os seis bits serão alterados para 100011, a fim de indicar que o pacote deve ser confirmado para *F*, mas não deve ser enviado para lá.

[arte: ver original p. 365]

[T]Tabela

			Flags de envio			Flags de confirmação			
Origem	Seq.	Idade	A	C	F	A	C	F	Dados
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

[F]Figura 5.14

[FL] O buffer de pacotes correspondente ao roteador *B* da Figura 5.13

[T4] Como calcular as novas rotas

Uma vez que uma rota tenha acumulado um conjunto completo de pacotes de estado de enlace, ela poderá criar o grafo da sub-rede completo, pois todo enlace estará representado. Na verdade, todo enlace será representado duas vezes, uma vez em cada sentido. Pode-se calcular a média dos dois valores ou usá-los separadamente.

Agora, o algoritmo de Dijkstra pode ser executado no local com a finalidade de criar o caminho mais curto até todos os destinos possíveis. Os resultados desse algoritmo podem ser instalados nas tabelas de roteamento, e a operação normal pode ser retomada.

No caso de uma sub-rede com *n* roteadores, cada qual com *k* vizinhos, a memória necessária para armazenar os dados de entrada é proporcional a *kn*. No caso de sub-redes de grande porte, isso pode ser um problema. Além disso, o tempo de cálculo também pode ser de grande importância. Contudo, em muitas situações práticas, o roteamento por estado de enlace funciona muito bem. Entretanto, problemas com o hardware ou com o software podem causar grandes complicações com esse algoritmo (e também com outros). Por exemplo, se um roteador alegar ter uma linha que na realidade não tem, ou esquecer uma linha que tem, o grafo da sub-rede ficará incorreto. Se um roteador deixar de encaminhar pacotes ou danificá-los enquanto os encaminhar, haverá problemas. Por fim, se a memória do roteador se esgotar ou se ele calcular o roteamento incorretamente, as falhas serão inúmeras. À medida que a sub-rede crescer até a faixa de dezenas ou centenas de milhares de nós, a probabilidade de algum roteador falhar ocasionalmente deixará de ser desprezível. O truque é tentar limitar os danos quando acontecer o inevitável. Perlman (1988) analisa em detalhes esses problemas e suas soluções.

O roteamento por estado de enlace é amplamente utilizado em redes reais; portanto, vale a pena fazer alguns comentários sobre alguns exemplos de protocolos que o utilizam. O protocolo OSPF, amplamente utilizado na Internet, emprega um algoritmo de estado de enlace. O OSPF será descrito na Seção 5.6.4. Outro protocolo de estado de enlace é o **IS-IS (Intermediate System-Intermediate System — sistema intermediário-sistema intermediário)**, projetado para a DECnet e adotado mais tarde pela ISO para uso com seu protocolo da camada de rede

sem conexões, o CLNP. Desde então, ele foi modificado para tratar de outros protocolos também; dentre eles, destacamos o IP. O protocolo IS-IS é utilizado em alguns backbones da Internet (incluindo o antigo backbone NSFNET) e em alguns sistemas celulares digitais como o CDPD. O Novell NetWare utiliza uma variante secundária do IS-IS (NLSP) para rotear pacotes IPX.

Basicamente, o IS-IS distribui uma visão instantânea da topologia de roteador, a partir da qual são calculados os caminhos mais curtos. Cada roteador anuncia, em suas informações de estado de enlace, quais endereços da camada de rede ele pode alcançar diretamente. Esses endereços podem ser IP, IPX, AppleTalk ou quaisquer outros endereços. O IS-IS é capaz até mesmo de aceitar vários protocolos da camada de rede ao mesmo tempo.

Muitas das inovações projetadas para o IS-IS foram adotadas pelo OSPF (o OSPF foi desenvolvido vários anos depois do IS-IS). Dentre essas inovações estão as seguintes: um método de autoestabilização de atualizações de estado de enlace por inundação, o conceito de um roteador designado em uma LAN, e o método de cálculo e suporte de divisão de caminhos, além de várias unidades métricas. Conseqüentemente, há pouca diferença entre o IS-IS e o OSPF. A mais importante delas é que o IS-IS é codificado de tal forma que se torne simples e natural transportar simultaneamente informações sobre vários protocolos da camada de rede, um recurso de que o OSPF não apresenta. Essa vantagem é especialmente valiosa em grandes ambientes de vários protocolos.

[T3] 5.2.6 Roteamento hierárquico

À medida que as redes aumentam de tamanho, as tabelas de roteamento dos roteadores crescem proporcionalmente. Não apenas a memória do roteador é consumida por tabelas cada vez maiores, mas também é necessário dedicar maior tempo da CPU para percorrê-las e mais largura de banda para enviar relatórios de

status sobre elas. Em um determinado momento, a rede pode crescer até o ponto em que deixará de ser viável cada roteador ter uma entrada correspondente a cada outro roteador, então o roteamento terá de ser feito de forma hierárquica, como na rede telefônica.

Quando o roteamento hierárquico for utilizado, os roteadores serão divididos naquilo que denominaremos **regiões**, com cada roteador conhecendo todos os detalhes sobre como rotear pacotes para destinos dentro de sua própria região, mas sem conhecer nada sobre a estrutura interna de outras regiões. Quando diferentes redes estão interconectadas, é natural que cada uma seja vista como uma região separada, a fim de liberar os roteadores de uma rede da necessidade de conhecer a estrutura topológica das outras redes.

No caso de redes muito grandes, uma hierarquia de dois níveis talvez seja insuficiente; provavelmente será necessário reunir as regiões em agrupamentos (clusters), os agrupamentos em zonas, as zonas em grupos etc., até faltarem nomes para os agregados. Como exemplo de uma hierarquia de vários níveis, vejamos como um pacote poderia ser roteado de Berkeley, na Califórnia, até Malindi, no Quênia. O roteador de Berkeley conheceria a topologia detalhada da Califórnia, mas enviaria todo o tráfego de fora do estado para o roteador de Los Angeles. O roteador de Los Angeles seria capaz de rotear o tráfego para outros roteadores domésticos, mas enviaria todo o tráfego destinado a outros países para Nova York. O roteador de Nova York seria programado de modo a direcionar todo o tráfego para o roteador do país de destino responsável pelo tratamento do tráfego vindo do exterior, digamos, em Nairóbi. Por fim, o pacote seguiria seu caminho descendente pela árvore no Quênia até chegar a Malindi.

A Figura 5.15 fornece um exemplo quantitativo do roteamento em uma hierarquia de dois níveis com cinco regiões. A tabela de roteamento completa do roteador 1*A* tem 17 entradas, como mostra a Figura 5.15(b). Quando o roteamento for

feito hierarquicamente, como na Figura 5.15(c), haverá entradas para todos os roteadores locais como antes, mas todas as outras regiões terão sido condensadas em um único roteador; portanto, todo o tráfego destinado à região 2 passará pela linha 1*B*–2*A*, mas o restante do tráfego remoto utilizará a linha 1*C*–3*B*. O roteamento hierárquico reduz a tabela de 17 para 7 entradas. À medida que a relação entre o número de regiões e o número de roteadores por região cresce, a economia de espaço na tabela aumenta.

[arte: ver original p. 367]

[Dísticos]

[1] Região 1

[2] Região 2

[3] Região 3 Região 4 Região 5

(a)

[4] Tabela completa para 1A

Dest. Linha Hops

[5] Tabela hierárquica para 1A

Dest. Linha Hops

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.15

[FL] Roteamento hierárquico

Infelizmente, esses ganhos em espaço não são gratuitos. Há um preço a ser pago, e esse preço tem a forma de um aumento no comprimento do caminho. Por exemplo, a melhor rota de 1*A* até 5*C* passa pela região 2; no entanto, com o roteamento hierárquico, todo o tráfego destinado à região 5 segue pela região 3,

porque essa é a melhor opção para a maior parte dos destinos da região 5.

Quando uma única rede se torna muito extensa, surge uma questão interessante: quantos níveis a hierarquia deve ter? Por exemplo, considere uma sub-rede com 720 roteadores. Se não houver hierarquia, cada roteador precisará de 720 entradas na tabela de roteamento. Se a sub-rede for particionada em 24 regiões de 30 roteadores cada uma, cada roteador precisará de 30 entradas locais e mais 23 entradas remotas, perfazendo um total de 53 entradas. Se for escolhida uma hierarquia de três níveis com oito agrupamentos, cada um deles contendo 9 regiões de 10 roteadores, cada roteador precisará de 10 entradas para roteadores locais, 8 entradas para roteamento até outras regiões dentro de seu próprio agrupamento e 7 entradas para agrupamentos distantes, perfazendo um total de 25 entradas. Kamoun e Kleinrock (1979) descobriram que o número ótimo de níveis para uma sub-rede com N roteadores é $\ln N$, exigindo um total de $e \ln N$ entradas por roteador. Eles também demonstraram que o aumento na extensão do caminho médio efetivo causado pelo roteamento hierárquico é suficientemente pequeno para ser aceitável de modo geral.

[T3] 5.2.7 Roteamento por difusão

Em algumas aplicações, os hosts precisam enviar mensagens a muitos outros hosts (ou a todos os outros hosts). Por exemplo, um serviço de distribuição de relatórios sobre o tempo, atualizações do mercado de ações ou programas de rádio ao vivo poderiam funcionar melhor pela difusão das informações a todas as máquinas, permitindo que aquelas que estivessem interessadas lessem os dados. O envio de um pacote a todos os destinos simultaneamente é chamado **difusão (broadcasting)**; foram propostos vários métodos para implementar esse recurso. Um método de difusão que não exige recursos especiais da sub-rede permite à origem simplesmente enviar um pacote específico a cada destino. O método não

só desperdiça largura de banda como também exige que a origem tenha uma lista completa de todos os destinos. Na prática, essa pode ser a única possibilidade. No entanto, esse é o menos desejável dos métodos.

O algoritmo de inundação é outro candidato óbvio. Ainda que o algoritmo de inundação seja inadequado para a comunicação comum ponto a ponto, ele pode ser levado em consideração para a difusão, especialmente se nenhum dos métodos descritos a seguir for aplicável. O problema da inundação como técnica de difusão é o mesmo problema que ela tem como um algoritmo de roteamento ponto a ponto: gera pacotes demais e consome largura de banda em excesso.

Um terceiro algoritmo é o **roteamento para vários destinos**. Se esse método for utilizado, cada pacote conterá uma lista de destinos ou um mapa de bits indicando os destinos desejados. Quando um pacote chega a um roteador, este verifica todos os destinos para determinar o conjunto de linhas de saída que serão necessárias. (Uma linha de saída será necessária se for a melhor rota para pelo menos um dos destinos.) O roteador gera uma nova cópia do pacote para cada linha de saída a ser utilizada e inclui em cada pacote somente os destinos que vão usar a linha. Na verdade, o conjunto de destinos é particionado entre as linhas de saída. Após um número suficiente de hops, cada pacote transportará somente um destino e poderá ser tratado como um pacote normal. O roteamento para vários destinos é como utilizar pacotes endereçados separadamente, exceto pelo fato de, quando vários pacotes tiverem de seguir a mesma rota, um deles pagará toda a passagem, e os restantes viajarão de graça.

Um quarto algoritmo de difusão faz uso explícito da árvore de escoamento para o roteador que inicia a difusão — ou qualquer outra árvore de amplitude conveniente. Uma **árvore de amplitude** é um subconjunto da sub-rede que inclui todos os roteadores, mas não contém nenhum loop. Se cada roteador souber quais de suas linhas pertencem à árvore de amplitude, ele poderá copiar um

pacote de difusão de entrada em todas as linhas da árvore de amplitude, exceto aquela em que o pacote chegou. Esse método faz excelente uso da largura de banda, gerando o número mínimo absoluto de pacotes necessários para realizar essa tarefa. O único problema é que cada roteador deve ter conhecimento de alguma árvore de amplitude para que o método seja aplicável. Às vezes, essas informações estão disponíveis (por exemplo, com o roteamento por estado de enlace), mas às vezes não (por exemplo, no caso do roteamento com vetor de distância).

Nosso último algoritmo de difusão é uma tentativa de aproximação com o comportamento do algoritmo anterior, mesmo quando os roteadores nada sabem sobre árvores de amplitude. A idéia, chamada **encaminhamento pelo caminho inverso** é muito simples, depois que é compreendida. Quando um pacote de difusão chega a um roteador, o roteador verifica se o pacote chegou pela linha que normalmente é utilizada para o envio de pacotes à origem da difusão. Em caso afirmativo, há uma excelente possibilidade de que o pacote de difusão tenha seguido a melhor rota a partir do roteador e seja, portanto, a primeira cópia a chegar ao roteador. Se for esse o caso, o roteador encaminhará cópias do pacote para todas as linhas, exceto aquela por onde ele chegou. No entanto, se o pacote de difusão tiver chegado em uma linha diferente da preferencial para alcançar a origem, ele será descartado como uma provável duplicata.

[arte: ver original p. 369]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.16

[FL] Encaminhamento pelo caminho inverso. (a) Uma sub-rede. (b) Uma árvore de escoamento. (c) A árvore construída por encaminhamento pelo caminho inverso

Um exemplo do algoritmo de encaminhamento pelo caminho inverso é mostrado na Figura 5.16. A parte (a) mostra uma sub-rede, a parte (b) mostra uma árvore de escoamento para o roteador *I* dessa sub-rede, e a parte (c) mostra como funciona o algoritmo de encaminhamento pelo caminho inverso. No primeiro hop, *I* envia pacotes para *F*, *H*, *J* e *N*, como indica a segunda linha da árvore. Cada um desses pacotes chega ao caminho preferencial para *I* (supondo-se que o caminho preferencial acompanhe a árvore de escoamento) e é então indicado por um círculo em torno da letra. No segundo hop, são gerados oito pacotes, dois por cada um dos roteadores que receberam um pacote no primeiro hop. Por sua vez, todos os oito pacotes chegam a roteadores não visitados anteriormente, e cinco deles chegam ao longo da linha preferencial. Dos seis pacotes gerados no terceiro hop, somente três chegam pelo caminho preferencial (em *C*, *E* e *K*); os outros são duplicatas. Depois de cinco hops e 24 pacotes, a difusão termina, em comparação com quatro hops e 14 pacotes que haveria se a árvore de escoamento fosse seguida exatamente.

A principal vantagem do encaminhamento pelo caminho inverso é que ele é ao mesmo tempo razoavelmente eficiente e fácil de implementar. Ele não exige que os roteadores saibam nada sobre árvores de amplitude, nem têm o overhead de uma lista de destino ou um mapa de bits em cada pacote de difusão, como ocorre na estratégia de endereçamento para vários destinos. Ele também não requer nenhum mecanismo especial para interromper o processo, como é o caso do algoritmo de inundação (um contador de hops em cada pacote e um conhecimento prévio do diâmetro da sub-rede, ou então uma lista de pacotes já vistos por origem).

[T3] 5.2.8 Roteamento por multidifusão

Algumas aplicações exigem que processos amplamente separados funcionem reunidos em grupos; por exemplo, um grupo de processos que implementa um sistema de bancos de dados distribuídos. Nessas situações, muitas vezes é necessário que um processo envie uma mensagem a todos os outros membros do grupo. Se o grupo for pequeno, ele poderá simplesmente enviar a cada um dos outros membros uma mensagem ponto a ponto. Se o grupo for grande, essa estratégia se tornará dispendiosa. Às vezes, a difusão pode ser utilizada; no entanto, o uso da difusão para informar a 1000 máquinas de uma rede com um milhão de nós é ineficiente, porque a maioria dos receptores não está interessada na mensagem (ou, pior ainda, os receptores estão definitivamente interessados, mas não vêem a mensagem). Desse modo, precisamos de um meio para enviar mensagens a grupos bem definidos que têm um tamanho numericamente grande, mas que são pequenos em comparação com a rede como um todo.

O envio de uma mensagem a um desses grupos denomina-se **multidifusão (multicasting)** e seu algoritmo de roteamento é chamado **roteamento por multidifusão**. Nesta seção, descreveremos uma forma de realizar o roteamento por multidifusão. Para obter informações adicionais, consulte (Chu *et al.*, 2000; Costa *et al.*, 2001; Kasera *et al.*, 2000; Madruga e Garcia-Luna-Aceves, 2001; Zhang e Ryu, 2001).

A mulidifusão exige o gerenciamento de grupos. Será preciso usar algum método para criar e destruir grupos, e para permitir que os processos entrem e saiam de grupos. O modo como essas tarefas serão realizadas não interessa ao algoritmo de roteamento. O que interessa é que, quando um processo se associar a um grupo, ele informará seu host desse fato. É importante que os roteadores saibam quais de seus hosts pertencem a cada um dos grupos. Os hosts devem informar seus roteadores sobre alterações na associação a grupos, ou então os roteadores terão de consultar seus hosts periodicamente. De qualquer forma, os roteadores

ficam sabendo quais de seus hosts estão em cada um dos grupos. Os roteadores darão essa informação a seus vizinhos, e assim a informação se propagará pela sub-rede.

Para executar o roteamento por multidifusão, cada roteador calcula uma árvore de amplitude que engloba todos os outros roteadores da sub-rede. Por exemplo, na Figura 5.17(a), temos uma sub-rede com dois grupos, 1 e 2. Alguns roteadores estão associados a hosts que pertencem a um ou a ambos os grupos, como indica a figura. Uma árvore de amplitude correspondente ao roteador situado mais à esquerda é mostrada na Figura 5.17(b).

Quando um processo envia um pacote de multidifusão a um grupo, o primeiro roteador examina sua árvore de amplitude e a poda, removendo todas as linhas que não levam a hosts que são membros do grupo. No nosso exemplo, a Figura 5.17(c) mostra a árvore de amplitude do grupo 1 podada. Da mesma forma, a Figura 5.17(d) mostra a árvore de amplitude do grupo 2 podada. Os pacotes de multidifusão só são encaminhados ao longo da árvore de amplitude apropriada.

[arte: ver original p. 371]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.17

[FL] (a) Uma rede. (b) Uma árvore de amplitude correspondente ao roteador mais à esquerda. (c) Uma árvore de multidifusão correspondente ao grupo 1. (d) Uma árvore de multidifusão correspondente ao grupo 2

Existem vários métodos que podem ser usados para podar a árvore de amplitude. O mais simples pode ser usado se o roteamento por estado de enlace for empregado e cada roteador estiver ciente da topologia completa, inclusive de

quais hosts pertencem a cada um dos grupos. Em seguida, a árvore de amplitude pode ser podada, começando pela extremidade de cada caminho, seguindo em direção à raiz e removendo todos os roteadores que não pertencem ao grupo em questão.

Quando se emprega o roteamento com vetor de distância, é possível utilizar uma outra estratégia de poda. O algoritmo básico é o encaminhamento pelo caminho inverso. Entretanto, sempre que um roteador sem hosts interessados em um grupo específico e sem conexões para outros roteadores recebe uma mensagem de multidifusão relacionada a esse grupo, ele responde com uma mensagem PRUNE, informando ao transmissor que este não deve enviar mais mensagens de multidifusão para esse grupo. Quando um roteador sem membros de grupos entre seus hosts recebe tais mensagens em todas as suas linhas, ele também pode responder com uma mensagem PRUNE. Assim, a sub-rede será podada recursivamente.

Uma desvantagem potencial desse algoritmo é que ele é mal dimensionado para redes grandes. Suponha que uma rede tenha *n* grupos, cada qual com uma média de *m* membros. Para cada grupo, devem ser armazenadas *m* árvores de amplitudes podadas, perfazendo um total de *mn* árvores. Quando há muitos grupos grandes, é necessário um espaço de armazenamento considerável para armazenar todas as árvores.

Um projeto alternativo utiliza **árvores baseadas no núcleo** (Ballardie *et al.*, 1993). Aqui é calculada uma única árvore de amplitude por grupo, com a raiz (o núcleo) próxima ao centro do grupo. Para enviar uma mensagem de multidifusão, um host a envia ao núcleo, que então faz a multidifusão ao longo da árvore de amplitude. Embora essa árvore não seja ótima para todas as origens, a redução dos custos de armazenamento de *m* árvores para uma única árvore por grupo é uma economia importante.

[T3] 5.2.9 Roteamento para hosts móveis

Hoje em dia, milhões de pessoas têm computadores portáteis, e em geral desejam ler suas mensagens de correio eletrônico e acessar seus sistemas de arquivos normais onde quer que estejam. Esses hosts móveis criam uma nova complicação: antes de rotear um pacote para um host móvel, primeiro a rede precisa localizá-lo. A questão da incorporação de hosts móveis a uma rede é muito recente, mas nesta seção faremos um apanhado geral do assunto e forneceremos uma solução possível.

O modelo do mundo que os projetistas de redes normalmente utilizam é mostrado na Figura 5.18. Nessa figura, temos uma WAN que consiste em roteadores e hosts. Conectadas à WAN há LANs, MANs e células sem fios do tipo que estudamos no Capítulo 2.

[arte: ver original p. 372]

[Dísticos]

[1]Célula sem fio

[2]Agente local

LAN local

[3]MAN

[4]WAN

[5]Host móvel

Agente externo

LAN externa

[F]Figura 5.18

[FL] Uma WAN à qual estão conectadas LANs, MANs e células sem fios

Os hosts que nunca se movem são chamados estacionários. Eles estão

conectados à rede por fios de cobre ou fibra ópticas. Por outro lado, podemos distinguir dois outros tipos de hosts. Os hosts migrantes são basicamente hosts estacionários que se deslocam de um local fixo para outro de tempos em tempos, mas que utilizam a rede apenas quando estão fisicamente conectados a ela. Os hosts visitantes realmente utilizam seus computadores em trânsito e querem manter suas conexões à medida que se deslocam. Utilizaremos a expressão **hosts móveis** para designar as duas últimas categorias, isto é, todos os hosts que estão fora de suas bases e que ainda querem se manter conectados.

Partimos do princípio de que todos os hosts têm um **local inicial** permanente, que nunca muda. Os hosts também têm um endereço local permanente que pode ser usado para determinar seus locais iniciais, de modo análogo à forma como o número de telefone 1-212-5551212 indica que se trata dos Estados Unidos (código de país 1) e de Manhattan (212). O objetivo do roteamento em sistemas com hosts móveis é tornar possível o envio de pacotes a hosts móveis que estejam usando seus endereços locais e fazer os pacotes alcançarem esses hosts de forma eficiente, onde quer que eles possam estar. Evidentemente, o problema é localizá-los.

No modelo da Figura 5.18, o mundo é dividido (geograficamente) em pequenas unidades. Vamos chamá-las de áreas; normalmente, uma área é uma LAN ou célula sem fio. Cada área tem um ou mais **agentes externos**, processos que controlam todos os hosts móveis que visitam a área. Além disso, cada área tem um **agente local**, que controla os hosts cuja base se encontra na área, mas no momento que estão visitando outra área.

Quando um novo host entra em uma área, seja conectando-se a ela (por exemplo, ligando fisicamente seu computador à LAN), ou simplesmente percorrendo a célula, seu computador deve se registrar com o agente externo dessa área. O procedimento de registro normalmente funciona da seguinte forma:

1. Periodicamente, cada agente externo transmite um pacote anunciando sua existência e seu endereço. Um host móvel recém-chegado pode aguardar uma dessas mensagens; no entanto, se nenhuma mensagem chegar rápido o suficiente, o host móvel poderá transmitir (por difusão) um pacote com a mensagem: Há algum agente externo por aí?
2. O host móvel se registra com o agente externo, fornecendo seu endereço local, o endereço atual da camada de enlace de dados e algumas informações de segurança.
3. O agente externo entra em contato o agente local do host móvel e diz: Um de seus hosts está por aqui. A mensagem do agente externo para o agente local contém o endereço de rede do agente externo. A mensagem contém ainda as informações de segurança, a fim de convencer o agente local de que o host móvel realmente está lá.
4. O agente local examina as informações de segurança, que contêm um timbre de hora, para provar que foi gerado há alguns segundos. Se tudo estiver correto, o agente local diz ao agente externo para prosseguir.
5. Quando o agente externo obtém a confirmação do agente local, ele cria uma entrada em suas tabelas e informa ao host móvel que agora ele está registrado.

No caso ideal, quando um host deixa uma área, isso também deve ser anunciado para permitir o cancelamento do registro, mas muitos usuários desligam seus computadores abruptamente quando terminam de usá-los.

Quando um pacote é enviado a um host móvel, ele é roteado até a LAN local do host, pois é isso que o endereço informa que deve ser feito, como ilustra a etapa 1 da Figura 5.19. Nessa figura, o transmissor situado na área nordeste da cidade de Seattle deseja enviar um pacote a um host que normalmente se encontra no outro lado dos Estados Unidos, em Nova York. Pacotes enviados ao host móvel em sua LAN local situada em Nova York são interceptados pelo agente local

daquela cidade. Em seguida, o agente local consulta a nova localização (temporária) do host móvel e encontra o endereço do agente externo que está tratando do host móvel, situado em Los Angeles.

Depois, o agente local executa duas ações. Primeiro, ele encapsula o pacote no campo de carga útil de um pacote externo e envia este último ao agente externo (etapa 2 da Figura 5.19). Esse mecanismo é chamado tunneling; vamos examiná-lo com detalhes mais adiante. Depois de obter o pacote encapsulado, o agente externo remove o pacote original do campo de carga útil e o envia ao host móvel como um quadro de enlace de dados.

Em segundo lugar, o agente local diz ao transmissor que daí em diante ele deverá enviar pacotes ao host móvel encapsulando-os no campo de carga útil de pacotes explicitamente endereçados ao agente externo, em vez de enviá-los ao endereço local do host móvel (etapa 3). Os pacotes subseqüentes podem agora ser roteados diretamente para o host por intermédio do agente externo (etapa 4), ignorando por completo o local inicial.

[arte: ver original p. 374]

[Dísticos]

[1]1. O pacote é enviado ao endereço local do host móvel

[2]2. O pacote é enviado por tunneling ao agente externo

[3] 3. O transmissor recebe o endereço do agente externo

[4] 4. Pacotes subseqüentes são enviados por tunneling ao agente externo

[F]Figura 5.19

[FL] Roteamento de pacotes para hosts móveis

Os diversos esquemas propostos diferem em muitos aspectos. Primeiro, existe a questão da proporção desse protocolo que será executada pelos roteadores e pelos hosts e, nesse último caso, por qual camada nos hosts. Em segundo lugar,

em alguns esquemas, os roteadores espalhados ao longo do caminho registram endereços mapeados, de forma que possam interceptar e redirecionar o tráfego, mesmo antes que ele possa chegar ao local inicial do host. Em terceiro lugar, em alguns esquemas, cada visitante recebe um endereço temporário exclusivo; em outros, o endereço temporário se refere a um agente que manipula o tráfego para todos os visitantes.

Em quarto lugar, os esquemas diferem no modo como conseguem lidar com pacotes endereçados a um destino e que devem ser entregues a um destino diferente. Uma opção é alterar o endereço de destino e simplesmente retransmitir o pacote modificado. Outra alternativa é encapsular o pacote inteiro, incluindo o endereço local, no campo de carga útil de outro pacote enviado ao endereço temporário. Por fim, os esquemas diferem em seus aspectos de segurança. Em geral, quando um host ou um roteador recebe uma mensagem da forma "A partir de agora, envie todas as mensagens de correio de Carla para mim", ele pode ter algumas dúvidas sobre quem enviou a mensagem e se fazer isso é uma boa idéia. Diversos protocolos de hosts móveis são descritos e comparados em (Hac e Guo, 2000; Perkins, 1998a; Snoeren e Balakrishnan, 2000; Solomon, 1998; e ainda Wang e Chen, 2001).

[T3] 5.2.10 Roteamento em redes ad hoc

Vimos como realizar o roteamento quando os hosts são móvel, mas os roteadores são fixos. Um caso ainda mais extremo é aquele em que os próprios roteadores são móveis. Entre as possibilidades estão:

1. Veículos militares em um campo de batalha sem qualquer infra-estrutura.
2. Uma frota de navios no mar.
3. Trabalhos de emergência em calamidades que destroem a infra-estrutura.
4. Um grupo de pessoas com notebooks em uma área que não tem instalações

802.11.

Em todos esses casos e em outros, cada nó consiste em um roteador e um host, em geral no mesmo computador. Redes de nós que simplesmente estão próximas entre si são chamadas **redes ad hoc** ou **MANETs (Mobile Ad hoc NETworks).** Vamos agora examiná-las rapidamente. Você poderá encontrar mais informações em (Perkins, 2001).

O que torna as redes ad hoc diferentes das redes fisicamente conectadas é que todas as regras habituais a respeito de topologias fixas, vizinhos fixos e conhecidos, relacionamento fixo entre endereço IP e localização e outras são repentinamente abandonadas. Os roteadores podem ir e vir, ou aparecer em novos lugares de um momento para outro. Com uma rede fisicamente conectada, se um roteador tiver um caminho válido para algum destino, esse caminho continuará a ser válido indefinidamente (desde que não ocorra uma falha em algum lugar no sistema). No caso de uma rede ad hoc, a topologia pode se alterar o tempo todo, e assim o interesse e até mesmo a validade dos caminhos podem se alterar de modo espontâneo, sem qualquer aviso. É desnecessário dizer que essas circunstâncias tornam o roteamento em redes ad hoc bem diferente do roteamento nas redes equivalentes fixas.

Foram propostos diversos algoritmos de roteamento para redes ad hoc. Um dos mais interessantes é o algoritmo de roteamento **AODV (Ad hoc On-demand Distance Vector)** (Perkins e Royer, 1999). Trata-se de um algoritmo semelhante ao algoritmo de roteamento com vetor de distância de Bellman-Ford, mas adaptado para funcionar em um ambiente móvel e que leva em conta a largura de banda limitada e a baixa duração das baterias nesse ambiente. Outra característica incomum é que ele é um algoritmo por demanda, isto é, determina uma rota até algum destino apenas quando alguém deseja enviar um pacote a esse destino. Agora, vamos ver o que isso significa.

[T4] Descoberta de rota

Em qualquer instante, uma rede ad hoc pode ser descrita por um grafo dos nós (roteadores + hosts). Dois nós estão conectados (isto é, têm um arco entre eles no grafo) se podem se comunicar diretamente utilizando seus sinais de rádio. Tendo em vista que um dos dois pode ter um transmissor mais potente que o outro, é possível que *A* esteja conectado a *B*, mas que *B* não esteja conectado a *A*. Porém, por simplicidade, iremos supor que todas as conexões são simétricas. Também devemos observar que o simples fato de dois nós estarem dentro do alcance de rádio um do outro não significa que eles estão conectados. Pode haver edifícios, montanhas ou outros obstáculos que bloqueiem sua comunicação.

Para descrever o algoritmo, considere a rede ad hoc da Figura 5.20, em que um processo no nó *A* deseja enviar um pacote o nó *I*. O algoritmo AODV mantém uma tabela em cada nó, classificada por destino, fornecendo informações sobre esse destino, inclusive a que vizinho enviar pacotes para alcançar o destino. Vamos supor que *A* procure em sua tabela e não encontre uma entrada correspondente a *I*. Agora, ela tem de descobrir uma rota até *I*. Essa propriedade de descoberta de rotas apenas quando elas são necessárias é o que torna esse algoritmo "por demanda".

[arte: ver original p. 376]

[Dísticos]

[1] Alcance de difusão de A

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.20

[FL] (a) Alcance da difusão de *A*. (b) Depois de *B* e *D* receberem a difusão de *A*. (c)

Depois de *C*, *F* e *G* receberem a difusão de *A*. (d) Depois de *E*, *H* e *I* receberem a difusão de *A*. Os nós sombreados são novos destinatários. As setas mostram as rotas inversas possíveis

Para localizar *I*, *A* constrói um pacote especial ROUTE REQUEST e o transmite por difusão. O pacote alcança *B* e *D*, como ilustra a Figura 5.20(a). De fato, a razão para *B* e *D* estarem conectados a *A* no grafo é que eles podem receber comunicações de *A*. Por exemplo, *F* não é mostrado com um arco para *A*, porque não pode receber o sinal de rádio de *A*. Desse modo, *F* não está conectado a *A*.

O formato do pacote ROUTE REQUEST é mostrado na Figura 5.21. Ele contém os endereços de origem e destino, em geral seus endereços IP, que identificam que está procurando por quem. Ele também contém um campo *ID de solicitação*, um contador local mantido separadamente por cada nó e incrementado toda vez que um pacote ROUTE REQUEST é transmitido. Juntos, os campos *Endereço de origem* e *ID de solicitação* identificam de forma exclusiva o pacote ROUTE REQUEST, a fim de permitir que os nós descartem quaisquer duplicatas que venham a receber.

[arte: ver original p. 377]

[Dísticos]

[1] Endereço de origem

[2] ID de solicitação

[3] Endereço de destino

[4] Seqüência de origem #

[5] Seqüência de destino #

[6] Contagem de hops

[F]Figura 5.21

[FL] Formato de um pacote ROUTE REQUEST

Além do contador *ID de solicitação*, cada nó também mantém um segundo contador de seqüência, incrementado sempre que é enviado um pacote ROUTE REQUEST (ou uma resposta ao pacote ROUTE REQUEST de outro roteador). Seu funcionamento é um pouco semelhante ao de um clock, e ele é usado para identificar novas rotas a partir de rotas antigas. O quarto campo da Figura 5.21 é o contador de seqüência de *A*; o quinto campo é o valor mais recente do número de seqüência de *I* que *A* detectou (0, se ele nunca foi detectado). O uso desses campos será esclarecido em breve. O último campo, *Contagem de hops*, controlará o número de hops que o pacote efetuou. Ele é inicializado com o valor 0.

Quando um pacote ROUTE REQUEST chega a um nó (nesse caso, *B* e *D*), ele é processado nas seguintes etapas:

1. O par (*Endereço de origem*, *ID de solicitação*) é procurado em uma tabela de histórico local para verificar se essa solicitação já foi vista e processada. Se for uma duplicata, ela será descartada e o processamento se interromperá. Se ela não for uma duplicata, o par será inserido na tabela de histórico, para que duplicatas futuras possam ser rejeitadas, e o processamento continuará.
2. O receptor procura o destino em sua tabela de rotas. Se for conhecida uma nova rota até o destino, será transmitido de volta à origem um pacote ROUTE REPLY, informando como chegar ao destino (basicamente: Use-me). Nesse caso, a palavra *nova* significa que o *Número de seqüência de destino* armazenado na tabela de roteamento é maior que ou igual ao *Número de seqüência de destino* contido no pacote ROUTE REQUEST. Se ele for menor, isso quer dizer que a rota armazenada é mais antiga que a rota anterior que a origem tinha para o destino, e então a etapa 3 será executada.
3. Tendo em vista que o receptor não conhece uma nova rota para o destino, ele incrementa o campo *Contagem de hops* e retransmite o pacote ROUTE REQUEST.

Ele também extrai os dados do pacote e o armazena como uma nova entrada em sua tabela de rotas inversas. Essas informações serão utilizadas para construir a rota inversa, de forma que a resposta possa voltar à origem mais tarde. As setas na Figura 5.20 são usadas para construir a rota inversa. Também será inicializado um timer com a entrada de rota inversa inserida mais recentemente. Se ele expirar, a entrada será eliminada.

Nem *B* nem *D* sabem onde está *I*, e então cada um deles cria uma entrada de rota inversa apontando de volta para *A*, como mostram as setas na Figura 5.20, e transmite o pacote com o campo *Contagem de hops* definido como 1. A difusão de *B* alcança *C* e *D*. *C* cria uma entrada para ela em sua tabela de rotas inversas e a retransmite. Em contraste, *D* a rejeita como uma duplicata. De modo semelhante, a difusão de *D* é rejeitada por *B*. Porém, a difusão de *D* é aceita por *F* e *G* e armazenada, como mostra a Figura 5.20(c). Depois de *E*, *H* e *I* receberem a difusão, o pacote ROUTE REQUEST finalmente alcança um destino que sabe onde *I* está, ou seja, o próprio *I*, como ilustra a Figura 5.20(d). Observe que, embora tenhamos mostrado as difusões em três etapas discretas nesse caso, as difusões de nós diferentes não são coordenadas de nenhuma forma.

Em resposta à solicitação recebida, *I* constrói um pacote ROUTE REPLY, como mostra a Figura 5.22. Os campos *Endereço de origem*, *Endereço de destino* e *Contagem de hops* são copiados da solicitação recebida, mas o campo *Número de seqüência de destino* é tirado de seu contador na memória. O campo *Contagem de hops* é definido como 0. O campo *Duração* controla por quanto tempo a rota é válida. Esse pacote é transmitido por unidifusão (unicast) para o nó de onde veio o pacote ROUTE REQUEST; nesse caso, o nó *G*. Então, ele segue o caminho inverso até *D* e finalmente até *A*. Em cada nó, o campo *Contagem de hops* é incrementado, de forma que o nó possa ver a que distância se encontra do destino (*I*).

[arte: ver original p. 378]

[Dísticos]

[1]Endereço de origem

[2]Endereço de destino

[3]Seqüência de destino #

[4]Contagem de hops

[5]Duração

[F]Figura 5.22

[FL] Formato de um pacote ROUTE REPLY

Em cada nó intermediário no caminho de volta, o pacote é inspecionado. Ele é inserido na tabela de roteamento local como uma rota para *I* se uma ou mais das três condições a seguir é satisfeita:

1. Não é conhecida nenhuma rota para *I*.

2. O número de seqüência correspondente a *I* no pacote ROUTE REPLY é maior que o valor encontrado na tabela de roteamento.

3. Os números de seqüência são iguais, mas a nova rota é mais curta.

Desse modo, todos os nós na rota inversa aprendem a rota para *I* "por tabela", como um subproduto da descoberta de rota de *A*. Os nós que receberam o pacote ROUTE REQUEST original, mas que não estavam no caminho inverso (*B*, *C*, *E*, *F* e *H* no exemplo), descartarão a entrada da tabela de rotas inversas quando o timer associado expirar.

Em uma rede grande, o algoritmo gera muitas difusões, até mesmo para destinos que estão próximos. O número de difusões pode ser reduzido da maneira ilustrada a seguir. O campo *Prazo de validade* (*Time to live*) do pacote IP é inicializado pelo transmissor com o diâmetro esperado da rede e é decrementado em cada hop. Se ele alcançar 0, o pacote será descartado, em vez de ser

transmitido por difusão.

O processo de descoberta é então modificado como a seguir. Para localizar um destino, o transmissor envia um pacote ROUTE REQUEST com o campo *Prazo de validade* definido como 1. Se não houver nenhuma resposta dentro de um período de tempo razoável, um outro pacote será enviado, dessa vez com *Prazo de validade* definido como 2. As tentativas subseqüentes utilizarão 3, 4, 5 etc. Desse modo a pesquisa será realizada por tentativa, primeiro no local, e depois em anéis cada vez mais amplos.

[T4] Manutenção de rotas

Por ser possível mover ou desativar os nós, a topologia pode mudar espontaneamente. Por exemplo, na Figura 5.20, se *G* for desativado, *A* não perceberá que a rota que esteve usando para *I* (*ADGI*) não é mais válida. O algoritmo precisa ter a possibilidade de lidar com esse problema. Periodicamente, cada nó transmite por difusão uma mensagem *Hello*. Cada um de seus vizinhos deve responder a essa mensagem. Se não chegar nenhuma resposta, o transmissor saberá que aquele vizinho saiu de seu alcance e não está mais conectado a ele. De modo semelhante, se ele tentar enviar um pacote a um vizinho que não responde, o nó aprenderá que o vizinho não está mais disponível.

Essas informações são usadas para purgar rotas que não estão mais ativas. Para cada destino possível, cada nó *N* mantém o controle de seus vizinhos que lhe enviaram um pacote para esse destino durante os últimos ΔT segundos. Esses vizinhos são chamados **vizinhos ativos** de *N* para esse destino. *N* faz isso mantendo uma tabela de roteamento organizada por destino e contendo o nó de saída a ser utilizado para acessar o destino, a contagem de hops até o destino, o número de seqüência de destino mais recente e a lista de vizinhos ativos

correspondente a esse destino. Uma tabela de roteamento possível para o nó *D* em nosso exemplo de topologia é mostrada na Figura 5.23(a).

[arte: ver original p. 379]

[Dísticos]

[1]

Dest.	Próximo hop	Distância	Vizinhos ativos	Outros campos
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.23

[FL] (a) Tabela de roteamento de *D* antes de *G* ficar inativo. (b) O grafo depois que *G* fica inativo

Quando qualquer dos seus vizinhos se torna inacessível, *N* verifica sua tabela de roteamento para ver quais destinos têm rotas que utilizam o vizinho agora inativo. Para cada uma dessas rotas, cada vizinho ativo é informado de que sua rota que passa por *N* agora é inválida e, portanto, deve ser purgada de suas tabelas de roteamento. Em seguida, os vizinhos ativos comunicam essa

informação a seus próprios vizinhos ativos e assim por diante, recursivamente, até todas as rotas que dependem do nó agora inativo serem retiradas de todas as tabelas de roteamento.

Como um exemplo de manutenção de rotas, considere nosso exemplo anterior, mas agora com *G* repentinamente desativado. A topologia alterada é ilustrada na Figura 5.23(b). Quando descobre que *G* está inativo, *D* observa sua tabela de roteamento e vê que *G* foi usado nas rotas para *E*, *G* e *I*. A união dos vizinhos ativos para esses destinos é o conjunto {*A*, *B*}. Em outras palavras, *A* e *B* dependem de *G* para algumas de suas rotas, e então têm de ser informados de que essas rotas não mais funcionam. *D* envia essa informação a eles trasmitindo pacotes que fazem com que esses vizinhos atualizem suas próprias tabelas de roteamento e, depois disso, *D* também purga as entradas para *E*, *G* e *I* de sua tabela de roteamento.

Talvez não tenha ficado óbvio em nossa descrição, mas uma diferença crítica entre AODV e Bellman-Ford é o fato de que os nós não trasmitem difusões periódicas contendo sua tabela de roteamento inteira. Essa diferença economiza largura de banda e também a carga das baterias.

AODV também é capaz de realizar o roteamento por difusão e por multidifusão. Para ver detalhes, consulte (Perkins e Royer, 2001). O roteamento ad hoc é uma área de pesquisa bastante intensa. Foram publicados muitos trabalhos sobre o assunto. Alguns desses documentos são (Chen *et al.*, 2002; Hu e Johnson, 2001; Li *et al.*, 2001; Raju e Garcia-Luna-Aceves, 2001; Ramanathan e Redi, 2002; Royer e Toh, 1999; Spohn e Garcia-Luna-Aceves, 2001; Tseng *et al.*, 2001; e Zadeh *et al.*, 2002).

[T3] 5.2.11 Pesquisa de nós em redes não hierárquicas

Um fenômeno relativamente novo é o das redes não hierárquicas, no qual um

grande número de pessoas, em geral com conexões físicas permanentes para a Internet, está em contato para compartilhar recursos. A primeira aplicação difundida da tecnologia não hierárquica foi utilizada para realizar um crime em massa: 50 milhões de usuários do Napster estavam trocando canções protegidas por direitos autorais sem a permissão dos proprietários desses direitos até o Napster ser fechado pela justiça em meio a grande controvérsia. Não obstante, a tecnologia não hierárquica tem muitos usos interessantes e legais. Ela também apresenta algo semelhante a um problema de roteamento, embora ele não seja exatamente igual aos problemas que estudamos até agora. Apesar disso, vale a pena um rápido exame.

O que torna os sistemas não hierárquicos interessantes é o fato de eles serem totalmente distribuídos. Todos os nós são simétricos e não existe nenhum controle ou hierarquia central. Em um sistema não hierárquico típico, cada um dos usuários tem alguma informação que pode ser de interesse para os outros usuários. Essas informações podem ser software gratuito, música (de domínio público), fotografias e assim por diante. Se houver grandes números de usuários, eles não conhecerão uns aos outros e não saberão onde encontrar o que estão procurando. Uma solução é um grande banco de dados central, mas isso talvez não seja viável por alguma razão (por exemplo, ninguém está disposto a hospedá-lo e mantê-lo). Desse modo, o problema se reduz a como um usuário pode encontrar um nó que contenha o que ele está procurando na ausência de um banco de dados centralizado ou mesmo de um índice centralizado.

Vamos supor que cada usuário tenha um ou mais itens de dados como canções, fotografias, programas, arquivos e assim por diante, que outros usuários talvez quissessem ler. Cada item tem um string ASCII para identificá-lo. Um usuário potencial conhece apenas o string ASCII e deseja descobrir se uma ou mais pessoas têm cópias e, se for o caso, quais são seus endereços IP.

Como um exemplo, considere um banco de dados distribuído de árvore genealógica. Cada estudioso de genealogia tem alguns registros on-line correspondentes a seus antepassados ou parentes, talvez com fotos, áudio, ou até mesmo videoclipes da pessoa. Várias pessoas podem ter o mesmo bisavô, ou um ancestral que deve ter registros em vários nós. O nome do registro é o nome da pessoa em alguma forma canônica. Em um certo momento, um estudioso de genealogia descobre o testamento de seu bisavô em um arquivo de armazenamento, no qual o bisavô deixa seu relógio de bolso de ouro como herança para seu sobrinho. Agora, o genealogista sabe o nome do sobrinho e quer descobrir se algum outro genealogista tem um registro correspondente a ele. Como descobrir quem tem algum registro, se for o caso, sem um banco de dados central?

Foram propostos diversos algoritmos para resolver esse problema. O que examinaremos é o algoritmo Chord (Dabek *et al.*, 2001a; e Stoica *et al.*, 2001). Uma explicação simplificada de como ele funciona é apresentada a seguir. O sistema Chord consiste em *n* usuários participantes, cada um dos quais pode ter alguns registros armazenados e está preparado para armazenar bits e itens do índice para utilização por outros usuários. Cada nó de usuário tem um endereço IP que pode ser mapeado para um número de *m* bits com a utilização de uma função de hash, denominada *hash*. O Chord emprega o SHA-1 na função *hash*. O SHA-1 é usado em criptografia; vamos estudá-lo no Capítulo 8. Por enquanto, ele é apenas uma função que toma como argumento um string de bytes de comprimento variável e produz um número de 160 bits altamente aleatório. Desse modo, podemos converter qualquer endereço IP em um número de 160 bits, chamado **identificador de nó.**

Conceitualmente, todos os 2^{160} identificadores de nós estão organizados em ordem ascendente em um grande círculo. Alguns correspondem a nós

participantes, embora a maioria deles não corresponda. Na Figura 5.24(a), mostramos o círculo de identificadores de nós para $m = 5$ (por enquanto, ignore os arcos intermediários). Nesse exemplo, os nós com identificadores 1, 4, 7, 12, 15, 20 e 27 correspondem a nós reais e estão sombreados na figura; os restantes não existem.

Vamos definir agora a função *sucessor*(*k*) como o identificador de nó do primeiro nó real seguinte a *k* em torno do círculo, no sentido horário. Por exemplo, *sucessor*(6) = 7, *sucessor*(8) = 12 e *sucessor*(22) = 27.

Os nomes dos registros (nomes de canções, nomes de antepassados e assim por diante) também são mapeados para números com a função *hash* (isto é, SHA-1) para gerar um número de 160 bits, denominado **chave**. Desse modo, para converter *nome* (o nome ASCII do registro) em sua chave, usamos *chave* = *hash*(*nome*). Esse cálculo é apenas uma chamada de procedimento local para *hash*. Se uma pessoa que possui um registro genealógico correspondente a *nome* quiser torná-lo disponível para todos, primeiro ela criará uma tupla do tipo (*nome*, *meu-endereço-IP*) e depois pedirá a *sucessor*(*hash*(*nome*)) para armazenar a tupla. Se existirem vários registros (em diferentes nós) correspondentes a esse nome, todas as suas tuplas serão armazenadas no mesmo nó. Desse modo, o índice será distribuído sobre os nós ao acaso. Para manter a tolerância a falhas, *p* funções de hash diferentes poderiam ser usadas para armazenar cada tupla em *p* nós, mas não consideraremos mais detalhes neste momento.

[arte: ver original p. 382]

[Dísticos]

[1]Nó real

[2]Identificador de nó

[3]Início Ender. IP do sucessor

Tabela finger do nó 1

[4]Início Ender. IP do sucessor

Tabela finger do nó 4

[5]Início Ender. IP do sucessor

Tabela finger do nó 12

(b)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.24

[FL] (a) Um conjunto de 32 identificadores de nós organizados em um círculo. Os números sombreados correspondem a máquinas reais. Os arcos mostram os fingers dos nós 1, 4 e 12. Os rótulos dos arcos são os índices das tabelas. (b) Exemplos das tabelas finger

Se mais tarde algum usuário quiser procurar *nome*, ele utilizará o hash do nome para obter *chave*, e depois usará *sucessor*(*chave*) para encontrar o endereço IP do nó que armazena suas tuplas de índice. A primeira etapa é fácil, mas a segunda não é. Para tornar possível encontrar o endereço IP do nó correspondente a uma certa chave, cada nó deve manter certas estruturas de dados administrativas. Ums delas é o endereço IP de seu nó sucessor ao longo do círculo de identificadores de nós. Por exemplo, na Figura 5.24, o sucessor do nó 4 é 7, e o sucessor do nó 7 é 12.

Agora, a pesquisa pode continuar da maneira descrita a seguir. O nó solicitante envia um pacote a seu sucessor contendo seu endereço IP e a chave que está procurando. O pacote se propaga pelo anel até localizar o sucessor para o identificador que está sendo procurado. Esse nó verifica se tem alguma

informação que corresponda à chave e, em caso afirmativo, devolve as informações diretamente ao solicitante, de cujo endereço IP ele dispões.

Como uma primeira otimização, cada nó poderia conter os endereços IP de seu sucessor e de seu predecessor, de forma que as consultas pudessem ser enviadas no sentido horário ou anti-horário, dependendo do pecurso que fosse considerado mais curto. Por exemplo, o nó 7 da Figura 5.24 poderia seguir o sentido horário para encontrar o identificador de nó 10, mas seguiria o sentido anti-horário para encontrar o identificador de nó 3.

Mesmo com duas opções de sentido, a pesquisa linear de todos os nós é muito ineficiente em um sistema não hierárquico de grande porte, pois o número médio de nós exigidos por pesquisa é $n/2$. Para acelerar bastante a procura, cada nó também mantém aquilo que o Chord denomina uma **tabela finger.** A tabela finger tem m entradas, indexadas por 0 a $m - 1$, cada uma apontando para um nó real diferente. Cada uma das entradas tem dois campos: *início* e o endereço IP de *sucessor*(*início*), como mostram os três exemplos de nós na Figura 5.24(b). Os valores dos campos correspondentes à entrada i no nó k são:

início $= k + 2^i$ (*módulo* 2^m)

Endereço IP de *sucessor*(*início*[*i*])

Observe que cada nó armazena os endereços IP de um número relativamente pequeno de nós, e que a maior parte desses nós tem identificadores de nós bastante próximos.

Utilizando a tabela finger, a pesquisa de *chave* no nó k prossegue da maneira ilustrada a seguir. Se *chave* ficar entre k e *sucessor*(k), então o nó que contém informações sobre *chave* é *sucessor*(k), e a pesquisa termina. Caso contrário, a tabela finger é pesquisada para encontrar a entrada cujo campo *início* é o predecessor mais próximo de *chave*. Então, uma solicitação é enviada diretamente ao endereço IP contido nessa entrada da tabela finger, solicitando

que ele continue a pesquisa. Tendo em vista que ele está mais próximo de *chave*, mas ainda se encontra abaixo dela, há boas chances de que ele seja capaz de retornar a resposta com apenas um pequeno número de consultas adicionais. De fato, tendo em vista que cada pesquisa reduz à metade a distância restante até o destino, é possível mostrar que o número médio de pesquisas é $\log_2 n$.

Como um primeiro exemplo, considere a pesquisa de *chave* = 3 no nó 1. Como o nó 1 sabe que 3 está entre ele e seu sucessor, 4, o nó desejado é 4 e a pesquisa termina, retornando o endereço IP do nó 4.

Como um segundo exemplo, considere a pesquisa de *chave* = 14 no nó 1. Como 14 não está entre 1 e 4, a tabela finger é consultada. O predecessor mais próximo a 14 é 9, e assim a solicitação é encaminhada ao endereço IP da entrada de 9, isto é, a do nó 12. O nó 12 verifica que 14 se encontra entre ele e seu sucessor (15), e assim retorna o endereço IP do nó 15.

Como um terceiro exemplo, considere a pesquisa de *chave* = 16 no nó 1. Novamente, é enviada uma consulta ao nó 12 mas, dessa vez, o nó 12 não sabe a resposta. Ele procura o nó mais próximo que antecede 16 e encontra 14, que gera o endereço IP do nó 15. Então, é enviada uma consulta para esse endereço. O nó 15 observa que 16 fica entre ele e seu sucessor (20), e então retorna o endereço IP do nó 20 ao chamador, e essa resposta volta até o nó 1.

Tendo em vista que os nós se juntam e se separam o tempo todo, Chord precisa de um meio para manipular essas operações. Supomos que, quando o sistema começou a operar, ele era pequeno o bastante para que os nós pudessem trocar informações diretamente, a fim de construir o primeiro círculo e as tabelas finger. Depois disso, é necessário um procedimento automatizado, conforme explicaremos a seguir. Quando um novo nó *r* quer se juntar a outros, ele deve entrar em contato com algum nó existente e pedir a ele que procure o endereço IP de *sucessor*(*r*) correspondente a ele. O novo nó procura *sucessor*(*r*) para seu

predecessor. O novo nó solicita ambos os endereços para inserir *r* entre eles no círculo. Por exemplo, se o nó 24 da Figura 5.24 quiser se juntar ao conjunto, ele pedirá que qualquer nó procure *sucessor*(24), que é 27. Em seguida, ele pergunta ao nó 27 por seu predecessor (20). Depois que o nó 24 informa ambos os nós sobre sua existência, 20 utiliza 24 como seu sucessor e 27 utiliza 24 como seu predecessor. Além disso, o nó 27 entrega as chaves no intervalo de 21 a 24, que agora pertencem ao nó 24. Nesse momento, 24 estará completamente inserido no conjunto de nós.

Porém, agora muitas tabelas finger estão incorretas. Para corrigi-las, todo nó executa um processo em background que recalcula periodicamente cada finger, chamando *sucessor*. Quando uma dessas consultas acessa um novo nó, a entrada da tabela finger correspondente é atualizada.

Quando um nó sai do conjunto de forma elegante, ele entrega suas chaves a seu sucessor e informa seu predecessor de sua partida, para que o predecessor possa se vincular ao sucessor do nó que deixou o conjunto. Quando um nó colidir, surgirá um problema, porque seu predecessor não terá mais um sucessor válido. Para atenuar esse problema, cada nó mantém o controle não apenas de seu sucessor direto, mas também de seus *s* sucessores diretos, a fim de permitir que ele possa ignorar até *s* – 1 nós defeituosos consecutivos e reconectar o círculo.

O sistema Chord foi usado para construir um sistema de arquivos distribuído (Dabek *et al.*, 2001b) e outras aplicações, e a pesquisa sobre seu uso é contínua. Um sistema não hierárquico diferente, denominado Pastry, e suas aplicações são descritos em (Rowstron e Druschel, 2001a; e Rowstron e Druschel, 2001b). Um terceiro sistema não hierárquico, chamado Freenet, é discutido em (Clarke *et al.*, 2002). Um quarto sistema desse tipo é descrito em (Ratnasamy *et al.*, 2001).

[T2] 5.3 Algoritmos de controle de congestionamento

Quando há pacotes demais presentes em (parte de) uma sub-rede, o desempenho diminui. Essa situação é chamada **congestionamento**. A Figura 5.25 ilustra o sintoma. Quando o número de pacotes depositados na sub-rede pelos hosts está dentro de sua capacidade de transporte, eles são todos entregues (exceto alguns que sofram com erros de transmissão), e o número entregue é proporcional ao número enviado. Entretanto, quando o tráfego aumenta muito, os roteadores já não são capazes de suportá-lo e começam a perder pacotes. Isso tende a piorar a situação. No caso de tráfego muito intenso, o desempenho entra em colapso total, e quase nenhum pacote é entregue.

[arte: ver original p. 385]

[Dísticos]

[1]Pacotes entregues

[2]Capacidade máxima de transporte da sub-rede

[3] Perfeita

[4] Desejável

[5] Congestionada

[6] Pacotes enviados

[F]Figura 5.25

[FL] Quando é oferecido tráfego demais, há congestionamento, e o desempenho diminui nitidamente

O congestionamento pode ser causado por diversos fatores. Se os fluxos de pacotes começarem a chegar repentinamente em três ou quatro linhas de entrada e todas precisarem da mesma linha de saída, haverá uma fila. Se a memória for insuficiente para conter todos eles, os pacotes se perderão. A inclusão de mais memória ajudará até certo ponto, mas Nagle (1987) descobriu que, se os roteadores tiverem um volume infinito de memória, o congestionamento piorará,

e não melhorará pois, no momento em que os pacotes chegarem ao início da fila, eles já terão sido temporizados (repetidamente) e as duplicatas já terão sido enviadas. Todos esses pacotes serão encaminhados com todo cuidado ao roteador seguinte, aumentando a carga até o destino.

Processadores lentos também podem causar congestionamento. Se as CPUs dos roteadores forem lentas na execução de tarefas administrativas (enfileiramento de buffers, atualização de tabelas etc.), poderão surgir filas, mesmo que haja capacidade de linha suficiente. Da mesma forma, linhas de baixa largura de banda também podem causar congestionamento. A atualização das linhas sem a alteração dos processadores, ou vice-versa, sempre ajuda um pouco, mas com freqüência transfere o gargalo. Além disso, a atualização parcial, e não integral, do sistema, costuma transferir o gargalo para outro lugar. Com freqüência, o problema real é uma incompatibilidade entre partes do sistema. Esse problema persistirá até que todos os componentes estejam em equilíbrio.

Vale a pena destacar explicitamente a diferença entre controle de congestionamento e controle de fluxo, pois o relacionamento entre eles é sutil. O controle de congestionamento se baseia na garantia de que a sub-rede é capaz de transportar o tráfego oferecido. É uma questão global, envolvendo o comportamento de todos os hosts, de todos os roteadores, do processamento de operações store-and-forward dentro dos roteadores e de todos os outros fatores que tendem a reduzir a capacidade de transporte da sub-rede.

Por outro lado, o controle de fluxo se baseia no tráfego ponto a ponto entre um determinado transmissor e um determinado receptor. Sua tarefa é garantir que um transmissor rápido não possa transmitir dados continuamente com maior rapidez do que o receptor é capaz de absorver. O controle de fluxo quase sempre envolve algum feedback direto do receptor para o transmissor. Dessa forma, o transmissor fica sabendo como tudo está sendo feito na outra extremidade.

Para perceber a diferença entre esses dois conceitos, considere uma rede de fibra óptica com capacidade de 1000 gigabits/s, na qual um supercomputador está tentando transferir um arquivo para um computador pessoal em uma velocidade de 1 Gbps. Mesmo que não haja congestionamento (a rede em si não apresenta problemas), o controle de fluxo é necessário para forçar o supercomputador a parar com freqüência, permitindo que o computador pessoal tenha a chance de "respirar".

No outro extremo, considere uma rede store-and-forward com linhas de 1 Mbps e 1000 computadores de grande porte, metade dos quais está tentando transferir arquivos a 100 kbps para a outra metade. O problema aqui não é o fato de os transmissores rápidos dominarem os receptores lentos, mas sim a questão do tráfego total oferecido exceder o que a rede é capaz de tratar.

A razão para o controle de congestionamento e o controle de fluxo serem confundidos com freqüência é que alguns algoritmos de controle de congestionamento operam enviando mensagens de volta às diversas origens, informando-as de que devem diminuir a velocidade quando a rede enfrentar problemas. Dessa forma, um host pode receber uma mensagem "reduzir velocidade", seja porque o receptor não pode manipular a carga, ou porque a rede não é capaz de tratá-la. Voltaremos a esse assunto mais adiante.

Vamos iniciar nosso estudo do controle de congestionamento examinando um modelo genérico para lidar com ele. Depois, veremos estratégias gerais para evitar os congestionamentos. Depois disso, analisaremos diversos algoritmos dinâmicos que permitem lidar com os congestionamentos, uma vez que eles se manifestem.

[T3] 5.3.1 Princípios gerais do controle de congestionamento

Muitos problemas em sistemas complexos, como as redes de computadores,

podem ser encarados do ponto de vista da teoria de controle. Essa abordagem nos leva à divisão de todas as soluções em dois grupos: loops abertos e loops fechados. As soluções para loops abertos tentam resolver o problema com um bom projeto, basicamente para garantir que eles em princípio não ocorrerão. Uma vez que o sistema esteja em funcionamento, não serão feitas correções que afetem processos em andamento.

Entre as ferramentas que controlam loops abertos, encontram-se as que são empregadas para decidir quando aceitar mais tráfego, para decidir quais pacotes devem ser descartados e quando isso deve ser feito, e ainda para programar decisões em vários pontos da rede. Tudo isso tem em comum o fato de que as decisões serão tomadas sem levar em conta o estado atual da rede.

Em contraste, as soluções para loops fechados se baseiam no conceito de um loop de feedback. Essa estratégia tem três partes, quando aplicada ao controle de congestionamento:

1. Monitorar o sistema para detectar quando e onde ocorre congestionamento.
2. Enviar essas informações para lugares onde alguma providência possa ser tomada.
3. Ajustar a operação do sistema para corrigir o problema.

Várias unidades métricas podem ser usadas para monitorar a sub-rede quanto à ocorrência de congestionamentos. As principais são a percentagem de todos os pacotes descartados por falta de espaço em buffer, a média dos comprimentos de fila, o número de pacotes interrompidos por alcançarem o tempo limite e que são retransmitidos, o retardo médio de pacotes e o desvio padrão do retardo de pacote. Em todos os casos, números crescentes indicam aumento de congestionamento.

A segunda etapa do loop de feedback é transferir informações sobre congestionamento do ponto em que o fenômeno é detectado para o ponto em

que algo pode ser feito em relação a ele. A solução mais óbvia é o roteador detectar o congestionamento com a finalidade de enviar um pacote à origem ou às origens de tráfego, anunciando o problema. É claro que esses pacotes extras aumentam a carga exatamente no momento em que não há necessidade de mais carga, ou seja, quando a sub-rede está congestionada.

Entretanto, também existem outras possibilidades. Por exemplo, pode-se reservar um bit ou um campo em todos os pacotes para que os roteadores o preencham sempre que o congestionamento superar algum limite inicial. Quando detecta esse estado de congestionamento, o roteador preenche o campo em todos os pacotes de saída, a fim de alertar os vizinhos.

Outra abordagem é fazer com que os hosts ou roteadores enviem pacotes de sondagem periodicamente para perguntar de forma explícita sobre o congestionamento. Essa informação pode então ser usada para rotear o tráfego em áreas problemáticas. Algumas estações de rádio têm helicópteros que voam sobre suas cidades com o objetivo de apresentar informações a respeito do congestionamentos nas estradas, tornando possível que seus ouvintes em trânsito façam o roteamento de pacotes (automóveis) de modo a evitar os pontos congestionados.

Em todos os esquemas de feedback, a esperança é que o conhecimento do congestionamento faça com que os hosts tomem as providências necessárias para reduzi-lo. Para um esquema desse tipo funcionar corretamente, a escala de tempo deve ser ajustada com cuidado. Se todas as vezes que dois pacotes chegarem em seqüência um roteador gritar PARE e toda vez que um roteador ficar ocioso por 20 μs ele gritar VÁ, o sistema oscilará muito e nunca convergirá. Por outro lado, se ele aguardar 30 minutos para ter certeza antes de comunicar algo, o mecanismo de controle de congestionamento reagirá muito lentamente para ter qualquer utilidade real. Para funcionar bem, é necessário utilizar algum

tipo de cálculo de média, mas obter a constante de tempo correta é uma questão não trivial.

Existem muitos algoritmos de controle de congestionamento. Para oferecer uma forma de organizá-los logicamente, Yang e Reddy (1995) desenvolveram uma taxonomia para algoritmos de controle de congestionamento. Eles começam dividindo todos os algoritmos em loops abertos ou em loops fechados, como descrevemos antes. Depois, dividem os algoritmos de loop aberto em algoritmos que atuam na origem e em algoritmos que atuam no destino. Os algoritmos de loop fechado também são divididos em duas subcategorias: feedback explícito e feedback implícito. Em algoritmos de feedback explícito, os pacotes são enviados do ponto de congestionamento para advertir a origem sobre o problema. Em algoritmos implícitos, a origem deduz a existência do congestionamento fazendo observações locais, como o tempo necessário para que as confirmações retornem.

A presença de congestionamento significa que a carga é (temporariamente) maior do que os recursos (de uma parte do sistema) podem manipular. Há duas soluções: aumentar os recursos ou diminuir a carga. Por exemplo, a sub-rede pode começar usando linhas telefônicas acessadas por discagem para aumentar temporariamente a largura de banda entre determinados pontos. Em sistemas de satélite, o aumento da potência de transmissão quase sempre proporciona largura de banda mais alta. A divisão do tráfego em várias rotas em vez de sempre utilizar a melhor rota também pode aumentar efetivamente a largura de banda. Por fim, roteadores sobressalentes que normalmente são utilizados apenas como reserva (para tornar o sistema tolerante a falhas) podem ser ativados para oferecer maior capacidade quando surgirem problemas sérios de congestionamento.

Entretanto, às vezes não é possível aumentar a capacidade, ou então ela já foi

aumentada até o limite máximo. A única forma de superar o congestionamento é diminuir a carga. Há diversas maneiras de reduzir a carga, incluindo negar o serviço a alguns usuários, piorar a qualidade do serviço para alguns ou para todos os usuários e fazer os usuários programarem suas necessidades de um modo mais previsível.

Alguns desses métodos, que estudaremos em breve, podem ser melhor aplicados a circuitos virtuais. No caso de sub-redes que utilizam circuitos virtuais internamente, esses métodos podem ser usados na camada de rede. No caso de sub-redes de datagramas, às vezes eles podem ser usados nas conexões no nível da camada de transporte. Neste capitulo, vamos nos concentrar em seu uso na camada de rede. No próximo capítulo, veremos o que pode ser feito na camada de transporte para administrar o congestionamento.

[T3] 5.3.2 Políticas de prevenção de congestionamento

Vamos começar nosso estudo de métodos de controle de congestionamento analisando os sistemas de loop aberto. Esses sistemas são projetados para minimizar antecipadamente o congestionamento, em vez de deixar que ele ocorra para depois reagir ao fato. Tais sistemas procuram atingir seus objetivos utilizando políticas apropriadas em vários níveis. Na Figura 5.26, vemos diferentes políticas para enlace de dados, rede e transporte que podem afetar o congestionamento (Jain, 1990).

[arte: ver original p. 388]

[T]Tabela

Camada	**Políticas**
Transporte	[B] Política de retransmissão
	[B] Política de cache fora de ordem
	[B] Política de confirmação

[B] Política de controle de fluxo

[B] Determinação de timeout

Rede [B] Circuitos virtuais *versus* datagramas na sub-rede

[B] Política de serviço e de enfileiramento de pacotes

[B] Política de descarte de pacotes

[B] Algoritmo de roteamento

[B] Gerenciamento da duração do pacote

Enlace de dados [B] Política de retransmissão

[B] Política de cache fora de ordem

[B] Política de confirmação

[B] Política de controle de fluxo

[F]Figura 5.26

[FL] Políticas que afetam o congestionamento

Começaremos pela camada de enlace de dados e, em seguida, passaremos às camadas superiores. A política de retransmissão trata da rapidez com que um transmissor chega ao timeout e do que ele transmite no timeout. Um transmissor dinâmico que chega ao timeout com rapidez e retransmite todos os pacotes pendentes usando go back n irá impor uma carga mais pesada sobre o sistema do que um transmissor lento que utilize retransmissão seletiva. Um fator intimamente relacionado a esse é a política de armazenamento em buffer. Se os receptores costumam descartar todos os pacotes fora da ordem, esses pacotes terão de ser retransmitidos mais tarde, criando carga extra. Com relação ao controle do congestionamento, a repetição seletiva é claramente melhor que a técnica de go back n.

A política de confirmação também afeta o congestionamento. Se cada pacote for confirmado imediatamente, os pacotes de confirmação irão gerar tráfego extra.

Entretanto, se as confirmações forem guardadas para serem transmitidas por piggyback sobre o tráfego inverso, poderão ocorrer interrupções e retransmissões extras. Um esquema de controle de fluxo rigoroso (por exemplo, uma janela pequena) reduz a taxa de dados e, portanto, ajuda a combater o congestionamento.

Na camada de rede, a escolha entre circuitos virtuais e datagramas afeta o congestionamento, pois muitos algoritmos de controle de congestionamento só funcionam com sub-redes de circuitos virtuais. A política de enfileiramento de pacotes e de serviços está relacionada ao fato de os roteadores terem uma fila por linha de entrada, uma fila por linha de saída ou os dois. Essa estratégia também está relacionada à ordem em que os pacotes são processados (por exemplo, por rodízio ou baseada na prioridade). A política de descarte é a regra que informa qual pacote será descartado quando não houver espaço. Uma boa política pode ajudar a aliviar o congestionamento, e uma política ruim pode torná-lo pior.

Um bom algoritmo de roteamento pode ajudar a evitar o congestionamento espalhando o tráfego por todas as linhas, enquanto uma política ruim pode enviar muito tráfego pelas linhas já congestionadas. Por fim, o gerenciamento no período de duração do pacote lida com o tempo de duração de um pacote antes de ser descartado. Se esse tempo for muito longo, os pacotes perdidos poderão atrapalhar o funcionamento por muito tempo; porém, se ele for breve demais, algumas vezes os pacotes poderão chegar ao timeout antes de alcançarem seu destino, induzindo assim as retransmissões.

Na camada de transporte, surgem as mesmas questões que ocorrem na camada de enlace de dados; no entanto, além disso, é mais difícil determinar o intervalo de timeout, porque o tempo de trânsito na rede é menos previsível que o tempo de trânsito sobre um fio entre dois roteadores. Se o inervalo de timeout for curto

demais, serão enviados pacotes extras desnecessariamente. Se ele for muito longo, o congestionamento será reduzido, mas o tempo de resposta será sacrificado sempre que um pacote for perdido.

[T3] 5.3.3 Controle de congestionamento em sub-redes de circuitos virtuais

Os métodos de controle de congestionamento que descrevemos são basicamente de loop aberto: eles tentam impedir que o congestionamento ocorra, em vez de lidar com ele após seu surgimento. Nesta seção, vamos descrever algumas estratégias para controlar dinamicamente o congestionamento em sub-redes de circuitos virtuais. Nas duas seções seguintes, vamos analisar técnicas que podem ser utilizadas em qualquer sub-rede.

Uma técnica amplamente utilizada para impedir que um congestionamento que já tenha começado se torne pior é o **controle de admissão.** A idéia é simples: uma vez que o congestionamento tenha dado alguma indicação de sua existência, nenhum outro circuito virtual será estabelecido até que o problema tenha passado. Portanto, todas as tentativas de estabelecer novas conexões da camada de transporte falharão. Admitir mais pessoas só irá piorar o problema. Apesar dessa estratégia ser um pouco grosseira, ela é simples e fácil de executar. No sistema telefônico, quando um switch fica sobrecarregado, o controle de admissão também é acionado, e não são fornecidos sinais de discagem.

Uma estratégia alternativa é permitir novos circuitos virtuais, mas rotear com cuidado todos os novos circuitos virtuais em áreas problemáticas. Considere a sub-rede da Figura 5.27(a), na qual dois roteadores estão congestionados.

[arte: ver original p. 390]

[Dísticos]

[1]A Congestionamento

B

Congestionamento

(a)

[2]A

Circuito virtual B

(b)

[F]Figura 5.27

[FL] (a) Uma sub-rede congestionada. (b) Uma sub-rede redesenhada que elimina o congestionamento. Também está representado um circuito virtual de *A* até *B*

Suponha que um host ligado ao roteador *A* queira estabelecer uma conexão com um host ligado ao roteador *B*. Normalmente, essa conexão passaria por um dos roteadores congestionados. Para evitar essa situação, podemos redesenhar a sub-rede da maneira ilustrada na Figura 5.27(b), omitindo os roteadores congestionados e todas as suas linhas. A linha tracejada mostra uma rota possível para o circuito virtual que evita os roteadores congestionados.

Outra estratégia relacionada a circuitos virtuais é negociar um acordo entre o host e a sub-rede quando um circuito virtual for configurado. Normalmente, esse acordo especifica o volume e a formatação do tráfego, a qualidade de serviço exigida e outros parâmetros. Para manter essa parte do acordo, em geral a sub-rede reservará recursos ao longo do caminho quando o circuito for configurado. Esses recursos podem incluir espaço em tabelas e buffers nos roteadores, além de largura de banda nas linhas. Dessa maneira, é improvável que ocorra congestionamento nos novos circuitos virtuais, pois todos os recursos necessários estarão disponíveis.

Esse tipo de reserva sempre poderá ser feito todo o tempo como procedimento de operação padrão ou somente quando a sub-rede estiver congestionada. Uma desvantagem de fazer isso o tempo todo é a tendência a desperdiçar recursos. Se

seis circuitos virtuais que poderiam utilizar 1 Mbps passarem todos pela mesma linha física de 6 Mbps, a linha terá de ser marcada como cheia, embora raramente aconteça de todos os seis circuitos virtuais estarem transmitindo na velocidade máxima ao mesmo tempo. Em conseqüência disso, o preço do controle de congestionamento é a largura de banda não utilizada (ou seja, desperdiçada) no caso normal.

[T3] 5.3.4 Controle do congestionamento em sub-redes de datagramas

Agora vamos examinar algumas estratégias que podem ser utilizadas em sub-redes de datagramas (e também em sub-redes de circuitos virtuais). Cada roteador pode monitorar facilmente a utilização de suas linhas de saída e de outros recursos. Por exemplo, o roteador pode associar a cada linha uma variável real, u, cujo valor entre 0,0 e 1,0 reflete a utilização recente dessa linha. Para manter uma boa estimativa de u, uma amostra da utilização instantânea da linha, f (0 ou 1), pode ser obtida periodicamente, sendo u atualizada de acordo com:

$$u_{nova} = a u_{antiga} + (1 - a) f$$

onde a constante a determina com que velocidade o roteador se esquece dos acontecimentos mais recentes.

Sempre que u ultrapassa o limite, a linha de saída entra em um estado de "advertência". Cada pacote recém-chegado é conferido para sabermos se sua linha de saída encontra-se em estado de advertência. Se estiver, alguma ação será adotada. A ação específica pode ser uma dentre as várias alternativas que examinaremos em seguida.

[T4] O bit de advertência

A antiga arquitetura DECNET assinalava o estado de advertência ativando um bit especial no cabeçalho do pacote. O mesmo é feito no frame relay. Quando o

pacote chegava a seu destino, a entidade de transporte copiava o bit na próxima confirmação a ser enviada de volta à origem. Em seguida, a origem interrompia o tráfego.

Enquanto estivesse no estado de advertência, o roteador continuava a definir o bit de advertência, e isso significava que a origem continuava a receber informações com o bit ativado. A origem monitorava a fração de confirmações com o bit ativado e ajustava sua velocidade de transmissão de acordo com ele. Enquanto os bits de advertência continuavam a fluir, a origem continuava a diminuir sua taxa de transmissão. Quando diminuía a velocidade de chegada das confirmações, a origem aumentava sua taxa de transmissão. Observe que, considerando que cada roteador ao longo do caminho podia ativar o bit de advertência, o tráfego só aumentava quando nenhum roteador tinha problemas.

[T4] Pacotes reguladores

O algoritmo de controle de congestionamento anterior é bastante sutil. Ele utiliza um modo cheio de rodeios para avisar à origem que ela deve transmitir mais lentamente. Por que não fazer isso diretamente? Segundo essa abordagem, o roteador enviará um **pacote regulador** ao host de origem, informando o destino encontrado no pacote. O pacote original é marcado (um bit de cabeçalho é ativado) para que ele não venha a gerar mais pacotes reguladores ao longo do caminho, e depois é encaminhado da forma habitual.

Ao receber o pacote regulador, o host de origem é obrigado a reduzir em *X*% o tráfego enviado ao destino especificado. Como outros pacotes com o mesmo destino provavelmente já estarão a caminho e irão gerar ainda mais pacotes reguladores, o host deve ignorar os pacotes reguladores que se refiram a esse destino por um intervalo de tempo fixo. Depois que esse tempo houver expirado, o host detectará mais pacotes reguladores para outro intervalo. Se chegar um

pacote, a linha ainda estará congestionada; logo, o host reduzirá o fluxo ainda mais e começará novamente a ignorar pacotes reguladores. Se não chegar nenhum outro pacote regulador durante o período de escuta, o host poderá aumentar o fluxo mais uma vez. O feedback implícito desse protocolo pode ajudar a evitar congestionamento sem estrangular o fluxo, a menos que ocorra algum problema.

Os hosts podem reduzir o tráfego ajustando seus parâmetros de orientação como, por exemplo, o tamanho de sua janela. Em geral, o primeiro pacote regulador faz a taxa de dados se reduzir a 0,50 de seu valor anterior, o seguinte causa uma redução de 0,25 e assim por diante. Os aumentos são feitos em incrementos menores para impedir que voltem rapidamente a ocorrer congestionamentos.

Foram propostas diversas variações desse algoritmo de controle de congestionamento. De acordo com uma delas, os roteadores podem manter diversos limiares. Dependendo do limiar que tiver sido ultrapassado, o pacote regulador poderá conter uma advertência suave, uma advertência severa ou um ultimato.

Outra variação é o uso de comprimentos de filas ou buffers, em vez de linhas como sinal para ativar o processo. É claro que é possível utilizar com essa métrica a mesma ponderação exponencial que foi usada com *u*.

[T4] Pacotes reguladores hop a hop

Em altas velocidades ou em longas distâncias, o envio de um pacote regulador para os hosts de origem não funciona bem porque a reação é muito lenta. Suponha que um host em San Francisco (o roteador *A* da Figura 5.28) esteja enviando tráfego para um host em Nova York (o roteador *D* da Figura 5.28) a 155 Mbps. Se o host de Nova York começar a esgotar o espaço de buffers, irá demorar

cerca de 30 ms para um pacote regulador voltar a San Francisco e solicitar que a transmissão seja mais lenta. A propagação do pacote regulador é mostrada como a segunda, terceira e quarta etapas da Figura 5.28(a). Nesses 30 ms, outros 4,6 megabits terão sido enviados. Mesmo que o host de San Francisco seja desativado por completo imediatamente, os 4,6 megabits do pipe continuarão a entrar em enxurrada e deverão ser processados. Somente no sétimo diagrama da Figura 5.28(a) o roteador de Nova York perceberá um fluxo mais lento.

Uma abordagem alternativa é fazer com que o pacote regulador tenha efeito a cada hop pelo qual passar, como mostra a seqüência da Figura 5.28(b). Aqui, assim que o pacote regulador atinge *F*, o nó *F* é solicitado a reduzir o fluxo para *D*. Fazendo isso, *F* terá de dedicar mais buffers ao fluxo, pois a origem ainda estará transmitindo a plena carga, mas dará alívio imediato a *D*, como um remédio para dor de cabeça em um comercial de televisão. Na etapa seguinte, o pacote regulador atingirá *E*, o que fará *E* reduzir o fluxo para *F*. Essa ação impõe uma demanda maior sobre os buffers de *E*, mas proporciona alívio imediato a *F*. Por fim, o pacote regulador atinge *A* e o fluxo diminui sua velocidade.

O efeito líquido desse esquema hop a hop é oferecer alívio rápido no ponto de congestionamento, ao preço de aumentar o consumo de buffers do fluxo ascendente (upstream). Dessa maneira, o congestionamento pode ser cortado pela raiz sem perda de pacotes. A idéia é analisada com mais detalhes, incluindo resultados de simulações, em (Mishra e Kanakia, 1992).

[arte: ver original p. 393]

[Dísticos]

[1] B C

A D

E F

[2] B C

A D

E F Fluxo intenso

[3]Regulador

Regulador

Regulador

Fluxo reduzido

[4]

Regulador

Regulador Fluxo reduzido

Regulador

[5]O fluxo ainda está na taxa máxima

O fluxo é reduzido

(a) (b)

[F]Figura 5.28

[FL] (a) Um pacote regulador que afeta apenas a origem. (b) Um pacote regulador que afeta cada hop por que passa

[T3] 5.3.5 Escoamento de carga

Quando nenhum dos métodos anteriores fizer o congestionamento desaparecer, os roteadores poderão chamar a artilharia pesada: o escoamento de carga. O **escoamento de carga** é uma maneira diferente de dizer que, quando os roteadores estão sendo inundados por pacotes que não podem manipular, eles simplesmente devem descartá-los. A expressão vem do universo da geração de energia elétrica, onde se refere à prática das concessionárias de apagar intencionalmente certas áreas para impedir que toda a rede entre em colapso nos dias quentes de verão, quando a demanda por eletricidade ultrapassa em muito a capacidade de abastecimento.

Um roteador que está sendo sufocado com pacotes pode simplesmente selecionar ao acaso aqueles que deverão ser descartados, mas em geral é possível fazer algo melhor que isso. O pacote a ser descartado pode depender das aplicações em execução. No caso de transferência de arquivos, um pacote antigo compensa mais que um novo, pois descartar o pacote 6 e manter os pacotes de 7 a 10 causará um intervalo no receptor que poderá forçar a retransmissão dos pacotes de 6 a 10 (se o receptor descartar rotineiramente pacotes fora de ordem). Em um arquivo de 12 pacotes, descartar o pacote 6 pode exigir a retransmissão dos pacotes de 7 a 12, enquanto a ação de descartar o pacote 10 deve exigir que apenas os pacotes de 10 a 12 sejam retransmitidos. Em comparação, no caso de multimídia, um novo pacote é mais importante que um antigo. A primeira dessas políticas (antigo é melhor que novo) costuma ser chamada política do **vinho**, e a segunda (novo é melhor que antigo) é chamada política do **leite**.

Uma etapa acima dessa em termos de inteligência exige a cooperação dos transmissores. No caso de muitas aplicações, alguns pacotes são mais importantes que outros. Por exemplo, certos algoritmos para compactação de vídeo transmitem periodicamente um quadro inteiro e depois enviam quadros subseqüentes sob a forma de diferenças em relação ao último quadro completo. Nesse caso, descartar um pacote que faz parte de uma diferença é mais interessante que descartar um pacote que faz parte de um quadro completo. Como outro exemplo, considere a transmissão de um documento contendo texto ASCII e figuras. A perda de uma linha de pixels em alguma imagem é muito menos prejudicial que a perda de uma linha de texto legível.

Para implementar uma política de descarte inteligente, as aplicações devem marcar seus pacotes por classes de prioridade, a fim de indicar qual a importância deles. Se as aplicações fizerem isso, quando os pacotes tiverem de ser descartados, os roteadores poderão descartar primeiro os pacotes da classe

mais baixa, depois os da classe mais baixa seguinte e assim por diante. É claro que, a menos que exista algum incentivo especial para marcar os pacotes com algo diferente de MUITO IMPORTANTE — NUNCA DESCARTAR, ninguém o fará.

O incentivo poderia vir sob a forma de dinheiro, com a transmissão dos pacotes de baixa prioridade sendo mais econômica que o envio dos pacotes de alta prioridade. Uma alternativa seria permitir que os transmissores enviassem pacotes de alta prioridade em condições de carga leve mas, à medida que a carga aumentasse, eles seriam descartados, encorajando assim os usuários a interromperem a transmissão desses quadros.

Uma outra opção é permitir que os hosts excedam os limites especificados no acordo negociado quando o circuito virtual foi configurado (por exemplo, usar uma largura de banda mais alta que a permitida), mas fiquem sujeitos à condição de que todo o tráfego em excesso seja marcado como tráfego de baixa prioridade. Na verdade, uma estratégia como essa não é má idéia, pois torna o uso de recursos ociosos mais eficiente, permitindo que os hosts os utilizem enquanto ninguém mais estiver interessado, mas sem estabelecer qualquer prioridade para eles quando houver disputa.

[T4] Detecção aleatória prematura

É bem conhecido o fato de que lidar com o congestionamento após sua detecção inicial é mais eficaz do que permitir que o congestionamento se consolide e depois tentar lidar com ele. Essa observação nos leva à idéia de descartar pacotes antes que todo o espaço dos buffers realmente se esgote. Um algoritmo popular para isso é chamado **RED (Random Early Detection — detecção aleatória prematura)** (Floyd e Jacobson, 1993). Em alguns protocolos de transporte (inclusive o TCP), a resposta à perda de pacotes é tornar a origem mais lenta. O raciocínio por trás dessa lógica é que o TCP foi projetado para redes fisicamente

conectadas e essas redes são muito confiáveis; assim, a perda de pacotes se deve muito mais ao estouro de buffers do que a erros de transmissão. Esse fato pode ser explorado para ajudar a reduzir o congestionamento.

Fazendo os roteadores descartarem pacotes antes que a situação se torne desesperadora (daí a palavra "prematura" no nome), a idéia consiste em ter tempo para empreender alguma ação antes de ser tarde demais. Para determinar quando começar a descartar pacotes, os roteadores mantêm uma média atualizada dos comprimentos de suas filas. Quando o comprimento médio da fila em alguma linha excede um limiar, considera-se que a linha está congestionada e é executada uma ação para solucionar o problema.

Tendo em vista que o roteador provavelmente não poderá detectar a origem que está causando a maior parte do problema, escolher um pacote ao acaso na fila que disparou a ação deve ser o melhor que ele consegue fazer.

Como o roteador deve fornecer informações à fonte sobre o problema? Uma alternativa é enviar um pacote regulador, conforme descrevemosantes. Porém, um problema que surge no caso dessa abordagem é que ela impõe ainda mais carga à rede já congestionada. Uma estratégia diferente é simplesmente descartar o pacote selecionado e não informar sobre o fato. A origem notará eventualmente a falta de confirmação e executará alguma ação. Como ela sabe que a perda de pacotes em geral é causada pelo congestionamento e por descartes, a origem responderá diminuindo a velocidade, em vez de continuar a tentar transmitir com maior intensidade. Essa forma implícita de feedback só funciona quando as origens respodem à perda de pacotes reduzindo sua taxa de transmissão. Em redes sem fios, nas quais a maioria das perdas se deve ao ruído no enlace aéreo, essa abordagem não pode ser usada.

[T3] 5.3.6 Controle de flutuação

Para aplicações como a transmissão de áudio e vídeo, não importa muito se os pacotes demoram 20 ms ou 30 ms para serem entregues, desde que o tempo em trânsito seja constante. A variação (isto é, o desvio padrão) nos tempos de chegada de pacotes é chamada **flutuação (jitter)**. Por exemplo, uma flutuação elevada, na qual alguns pacotes demoram 20 ms e outros demoram 30 ms para chegar, resultará em uma qualidade irregular do som ou do filme. A flutuação está ilustrada na Figura 5.29. Em contraste, um acordo em que 99% dos pacotes fossem entregues com um retardo no intervalo de 24,5 ms a 25,5 ms poderia ser aceitável.

É claro que o valor médio escolhido deve ser viável. Ele deve levar em conta o tempo de trânsito na velocidade da luz e o retardo mínimo na passagem pelos roteadores, e talvez deixar uma pequena folga para alguns retardos inevitáveis.

[arte: ver original p. 396]

[Dísticos]

[1] Fração de pacotes

[2] Alta flutuação

[3] Retardo mínimo (devido à velocidade da luz)

[4] Retardo

(a)

[5] Fração de pacotes

[6] Baixa flutuação

[7] Retardo

(b)

[F]Figura 5.29

[FL] (a) Alta flutuação. (b) Baixa flutuação

A flutuação pode ser limitada pelo cálculo do tempo de trânsito esperado para

cada hop ao longo do caminho. Quando um pacote chega a um roteador, este verifica se o pacote está adiantado ou atrasado em suas programação. Essas informações são armazenadas no pacote e atualizadas a cada hop. Se estiver adiantado, o pacote será retido um tempo suficiente para que seja sincronizado. Se ele estiver atrasado, o roteador tentará enviá-lo rapidamente.

Na realidade, o algoritmo que define, entre os diversos pacotes que disputam uma linha de saída, o pacote que deve ser enviado em seguida sempre pode escolher o pacote que estiver mais atrasado em sua programação. Dessa forma, pacotes que estão adiantados na programação têm sua velocidade reduzida, e pacotes que estão atrasados são acelerados; em ambos os casos, essa ação reduz o volume de flutuação.

Em algumas aplicações, como vídeo por demanda, a flutuação pode ser eliminada pelo armazenamento em buffer no receptor, seguido pela busca de dados para exibição no buffer, e não na rede em tempo real. No entanto, para outras aplicações, em especial aquelas que exigem interação em tempo real entre pessoas, como telefonia da Internet e videoconferência, o retardo inerente do armazenamento em buffer não é aceitável.

O controle de congestionamento é uma área de pesquisa ativa. O estado da arte é resumido em (Gevros *et al.*, 2001).

==================================

[T2] 5.4 Qualidade de serviço

As técnicas que examinamos nas seções anteriores foram projetadas para reduzir o congestionamento e melhorar o desempenho das redes. Porém, com o crescimento da multimídia em rede, freqüentemente essas medidas ad hoc não são suficientes. Há necessidade de empreender tentativas sérias para garantir a qualidade de serviço por meio do projeto de redes e protocolos. Nas próximas seções continuaremos nosso estudo do desempenho de rede, mas agora com um foco mais nítido sobre as alternativas para oferecer uma qualidade de serviço adequada às necessidades das aplicações. Porém, devemos observar que muitas dessas idéias ainda estão sendo desenvolvidas e podem sofrer modificações.

[T3] 5.4.1 Requisitos

Uma seqüência de pacotes desde uma origem até um destino é chamada **fluxo.** Em uma rede orientada a conexões, todos os pacotes que pertencem a um fluxo seguem a mesma rota; em uma rede sem conexões, eles podem seguir rotas diferentes. As necessidades de cada fluxo podem ser caracterizadas por quatro parâmetros principais: confiabilidade, retardo, flutuação e largura de banda. Juntos, esses parâmetros definem a **QoS (Quality of Service — qualidade de serviço)** que o fluxo exige. Várias aplicações comuns e a severidade de seus requisitos estão listadas na Figura 5.30.

[arte: ver original p. 397]

[T]Tabela

Aplicação	**Confiabilidade**	**Retardo**	**Flutuação**	**Largura de banda**
Correio eletrônico	Alta	Baixa	Baixa	Baixa
Transferência de arquivos	Alta	Baixa	Baixa	Média
Acesso à Web	Alta	Média	Baixa	Média

Login remoto Alta Média Média Baixa

Áudio por demanda Baixa Baixa Alta Média

Vídeo por demanda Baixa Baixa Alta Alta

Telefonia Baixa Alta Alta Baixa

Videoconferência Baixa Alta Alta Alta

[F]Figura 5.30

[FL] A rigidez dos requisitos de qualidade de serviço

As quatro primeiras aplicações têm requisitos estritos de confiabilidade. Nenhum bit pode ser entregue de forma incorreta. Em geral, esse objetivo é alcançado calculando-se o total de verificação de cada pacote e conferindo-se o total de verificação no destino. Se um pacote for danificado em trânsito, ele não será confirmado e será retransmitido mais tarde. Essa estratégia proporciona alta confiabilidade. As quatro últimas aplicações (áudio/vídeo) podem tolerar erros, e assim nenhum total de verificação é calculado ou conferido.

As aplicações de transferência de arquivos, incluindo correio eletrônico e vídeo, não são sensíveis ao retardo. Se todos os pacotes estiverem uniformemente atrasados alguns segundos, não haverá nenhum dano. Aplicações interativas, como navegação na Web e login remoto, são mais sensíveis ao retardo. Aplicações de tempo real, como telefonia e videoconferência, têm requisitos estritos de retardo. Se todas as palavras em uma ligação telefônica forem retardas exatamente 2,000 segundos, os usuários irão considerar a conexão inaceitável. Por outro lado, a reprodução de arquivos de áudio ou vídeo de um servidor não exige baixo retardo.

As três primeiras aplicações não são sensíveis à chegada de pacotes com intervalos de tempo irregulares entre eles. O login remoto é um pouco mais sensível a essa variação, pois os caracteres aparecerão na tela em pequenas

rajadas se a conexão sofrer muita flutuação. O vídeo e, em especial, o áudio são extremamente sensíveis à flutuação. Se um usuário estiver assistindo a um vídeo transmitido pela rede e os quadros estiverem todos atrasados exatamente 2,000 segundos, não haverá nenhum dano. Porém, se o tempo de transmissão variar ao acaso entre 1 e 2 segundos, o resultado será terrível. No caso do áudio, até mesmo uma flutuação de até alguns milissegundos será bastante audível.

Por fim, as aplicações diferem em suas necessidades de largura de banda. O correio eletrônico e o login remoto não necessitam de muita largura de banda, mas todas as formas de vídeo exigem um grande volume desse recurso.

As redes ATM classificam os fluxos em quatro categorias principais com relação às suas demandas de QoS:

1. Taxa de bit constante (por exemplo, telefonia).
2. Taxa de bits variável em tempo real (por exemplo, videoconferência compactada).
3. Taxa de bits variável não de tempo real (por exemplo, assistir a um filme pela Internet).
4. Taxa de bits disponível (por exemplo, transferência de arquivos).

Essas categorias também são úteis para outros propósitos e outras redes. A taxa de bits constante é uma tentativa de simular um fio oferecendo uma largura de banda uniforme e um retardo uniforme. A taxa de bits variável ocorre quando o vídeo é compactado, sendo alguns quadros mais compactados que outros. Desse modo, a transmissão de um quadro com uma grande quantidade de detalhes pode exigir o envio de muitos bits, enquanto a transmissão de uma foto de uma parede branca pode se compactar extremamente bem. A taxa de bits disponível se destina a aplicações como correio eletrônico, não sensíveis ao retardo ou à flutuação.

[T3] 5.4.2 Técnicas para se alcançar boa qualidade de serviço

Agora que sabemos algo sobre requisitos de QoS, como alcançá-los? Bem, para começar, não há nenhuma fórmula mágica. Nenhuma técnica isolada proporciona QoS eficiente e seguro de forma ótima. Em vez disso, foram desenvolvidas diversas técnicas, e as soluções práticas muitas vezes combinam várias dessas técnicas. Agora, examinaremos algumas técnicas que os projetistas de sistemas utilizam para alcançar QoS.

[T4] Superdimensionamento

Uma solução prática é fornecer tanta capacidade de roteadores, tanto de espaço de buffers e tanta largura de banda que os pacotes simplesmente são transmitidos com enorme facilidade. O problema com essa solução é seu custo. Com o passar do tempo e à medida que os projetistas adquirem uma idéia melhor da quantidade suficiente de recursos, essa técnica pode até mesmo se tornar prática. Até certo ponto, o sistema de telefonia é superdimensionado. É raro levantarmos o fone do gancho e não conseguirmos imediatamente um sinal de discagem. Nesse caso, há tanta capacidade disponível que a demanda sempre pode ser atendida.

[T4] Armazenamento em buffers

Os fluxos podem ser armazenados em buffers no lado receptor, antes de serem entregues. O armazenamento dos fluxos em buffers não afeta a confiabilidade ou a largura de banda e aumenta o retardo mas, por outro lado, suaviza a flutuação. No caso de áudio e vídeo por demanda, a flutuação é o principal problema e, portanto, essa técnica ajuda bastante.

Vimos a diferença entre alta flutuação e baixa flutuação na Figura 5.29. Na Figura 5.31, observamos uma série de pacotes sendo entregue com uma flutuação

significativa. O pacote 1 é enviado pelo servidor no tempo *t* = 0 segundo e chega ao cliente no tempo *t* = 1 segundo. O pacote 2 sofre maior retardo e demora 2 segundos para chegar. À medida que chegam, os pacotes são armazenados em buffers na máquina cliente.

[arte: ver original p. 399]

[Dísticos]

[1]O pacote deixa a origem 1 2 3 4 5 6 7 8

[2]O pacote chega ao buffer 1 2 3 4 5 6 7 8

[3]Pacote removido do buffer Tempo no buffer 1 2 3 4 5 6 7 8

Intervalo na reprodução

[4]0 5 10 15 20

Tempo (segundos)

[F]Figura 5.31

[FL] Suavizando o fluxo de saída por meio do armazenamento de pacotes em buffers

Em *t* = 10 s, inicia-se a reprodução. Nesse momento, os pacotes de 1 até 6 são armazenados no buffer, de forma que podem ser removidos do buffer em intervalos uniformes para reprodução suave. Infelizmente, o pacote 8 se atrasou tanto que não está disponível quando surge seu slot de reprodução, e assim a reprodução tem de parar até ele chegar, o que cria um incômodo intervalo na música ou no filme. Esse problema pode ser atenuado retardando-se ainda mais o momento do início, apesar disso também exigir um buffer maior. Todos os Web sites comerciais que contêm fluxo de áudio ou vídeo utilizam reprodutores que armazenam os itens em buffers por cerca de 10 segundos antes de iniciarem a reprodução.

[T4] Moldagem de tráfego

No exemplo anterior, a origem transmite os pacotes com um espaçamento uniforme entre eles mas, em outros casos, é possível que eles sejam transmitidos de modo irregular, o que pode causar congestionamentos na rede. A saída não uniforme é comum se o servidor está manipulando muitos fluxos ao mesmo tempo e ela também permite outras ações, como avanço e retrocesso rápidos, autenticação de usuários e assim por diante. Além disso, a abordagem que utilizamos aqui (de armazenamento em buffers) nem sempre é possível, por exemplo, com videoconferência. Porém, se fosse possível fazer algo para obrigar o servidor (e os hosts em geral) a transmitir a uma velocidade uniforme, a qualidade de serviços ria melhor. Examinaremos agora uma técnica denominada **moldagem de tráfego** que suaviza o tráfego no lado servidor, e não no lado cliente.

A moldagem de tráfego está relacionada à regulagem da *taxa* média (e do volume) da transmissão de dados. Por outro lado, os protocolos de janela deslizante estudados anteriormente limitam o volume de dados em trânsito de uma vez, e não a taxa em que eles são enviados. Quando uma conexão é configurada, o usuário e a sub-rede (isto é, o cliente e a concessionária de comunicações) concordam com um determinado padrão de tráfego (ou seja, uma forma) para esse circuito. Às vezes, esse acordo é chamado **acordo de nível de serviço**. Desde que o cliente cumpra sua parte no negócio e envie somente pacotes que estejam de acordo com o contrato, a concessionária de comunicações promete entregá-los pontualmente. A moldagem de tráfego reduz o congestionamento e ajuda a concessionária a cumprir sua promessa. Esses acordos não são muito importantes para transferências de arquivos, mas são de grande importância no caso da transmissão de dados em tempo real, como conexões de áudio e vídeo, que têm requisitos estritos de qualidade de serviço.

Na verdade, com a moldagem de tráfego, o cliente diz à concessionária de comunicações: "Meu padrão de transmissão ficará assim. Você pode lidar com ele?" Se a concessionária concordar, a questão será como a concessionária poderá saber se o cliente está cumprindo o acordo e o que fazer em caso negativo. O monitoramento de um fluxo de tráfego é chamado @@@controle de tráfego. Concordar com um padrão de tráfego e controlá-lo daí em diante é mais fácil com sub-redes de circuitos virtuais do que com sub-redes de datagramas. Entretanto, mesmo com sub-redes de datagramas, as mesmas idéias podem se aplicar a conexões da camada de transporte.

[T4] O algoritmo de balde furado

Imagine um balde com um pequeno furo no fundo, como ilustra a Figura 5.32(a). Independente da velocidade com que a água entra no balde, o fluxo de saída ocorrerá em uma taxa constante, ρ, quando houver qualquer quantidade de água no balde e zero quando o balde estiver vazio. Além disso, quando o balde estiver cheio, a água que entrar escorrerá pelas bordas e se perderá (ou seja, não aparecerá no fluxo de saída sob o furo).

A mesma idéia pode ser aplicada a pacotes, como mostra a Figura 5.32(b). Conceitualmente, cada host está conectado à rede por uma interface que contém um balde furado, ou seja, uma fila interna finita. Se um pacote chegar à fila quando ela estiver cheia, o pacote será descartado. Em outras palavras, se um ou mais processos dentro do host tentar enviar um pacote quando o número máximo já estiver enfileirado, o novo pacote será descartado sem cerimônia. Essa disposição pode ser interna à interface de hardware ou simulada pelo sistema operacional do host. Ela foi proposta primeiro por Turner (1986) e é chamada **algoritmo de balde furado.** Na verdade, trata-se simplesmente de um sistema de enfileiramento de um único servidor com um tempo de serviço constante.

[arte: ver original p. 401]

[Dísticos]

[1] Torneira

[2] Balde furado

[3] Água

[4] A água sai por um furo a uma velocidade constante

(a)

[5] Computador host

[6] Interface contendo um balde furado

[7] Pacote

Fluxo desregulado

O balde contém pacotes

Fluxo regulado

[8] Rede

(b)

[F]Figura 5.32

[FL] (a) Um balde furado com água. (b) Um balde furado com pacotes

O host pode inserir um pacote na rede a cada pulso de clock. Novamente, isso pode ser forçado pela placa de interface ou pelo sistema operacional. Esse mecanismo transforma um fluxo de pacotes irregular proveniente dos processos do usuário no host em um fluxo de pacotes regular para a rede, suavizando as rajadas e reduzindo bastante as possibilidades de congestionamento.

Quando os pacotes têm todos o mesmo tamanho (por exemplo, células ATM), esse algoritmo pode ser usado da maneira que descrevemos. Entretanto, quando estão sendo utilizados pacotes de tamanho variável, a melhor opção é permitir um número fixo de bytes por pulso, em vez de apenas um pacote. Assim, se a

regra for 1024 bytes por pulso, é possível admitir em um pulso um único pacote de 1024 bytes, dois pacotes de 512 bytes, quatro pacotes de 256 bytes etc. Se a contagem de bytes residuais for muito baixa, o próximo pacote deverá aguardar até o pulso seguinte.

É fácil implementar o algoritmo de balde furado original. O balde furado consiste em uma fila finita. Ao chegar um pacote, se houver espaço na fila, ele será incluído na fila; caso contrário, ele será descartado. A cada pulso do clock, um pacote é transmitido (a menos que a fila esteja vazia).

O balde furado de contagem de bytes é implementado praticamente da mesma maneira. A cada pulso, um contador é inicializado em *n*. Se o primeiro pacote da fila tiver menos bytes que o valor atual do contador, ele será transmitido, e o contador será decrementado por esse número de bytes. Também é possível enviar pacotes adicionais, desde que o contador tenha um valor suficientemente alto. Quando o contador fica abaixo do comprimento do próximo pacote na fila, a transmissão é interrompida até o pulso seguinte, quando então a contagem de bytes residuais é reinicializada e o fluxo pode continuar.

Como exemplo de um balde furado, imagine que um computador possa produzir dados a 25 milhões de bytes/s (200 Mbps) e que a rede também funcione nessa velocidade. Entretanto, os roteadores só podem lidar com essa taxa de dados durante intervalos curtos (basicamente até seus buffers serem preenchidos). No caso de intervalos longos, eles funcionam melhor a taxas que não excedem 2 milhões de bytes/s. Agora, suponha que os dados venham em rajadas de 1 milhão de bytes, e que ocorra uma rajada com a duração de 40 ms a cada segundo. Para reduzir a taxa média a 2 MB/s, poderíamos usar um balde furado com ρ = 2 MB/s e uma capacidade *C* igual a 1 MB. Isso significa que rajadas de até 1 MB podem ser manipuladas sem perda de dados e que tais rajadas se distribuem por 500 ms, independente da velocidade com que chegam.

Na Figura 5.33(a), vemos a entrada do balde furado funcionando a 25 MB/s por 40 ms. Na Figura 5.33(b), vemos a saída se esvaziando a uma taxa uniforme de 2 MB/s durante o período de 500 ms.

[T4] O algoritmo de balde de símbolos

O algoritmo de balde furado impõe um padrão de saída rígido à taxa média, independente da irregularidade do tráfego. Em muitas aplicações, é melhor permitir que a saída aumente um pouco sua velocidade quando chegarem rajadas maiores; assim, é necessário um algoritmo mais flexível, de preferência um que nunca perca dados. Um algoritmo como esse é o **algoritmo de balde de símbolos (token bucket algorithm).** Nesse algoritmo, o balde furado retém símbolos (ou tokens), gerados por um clock na velocidade de um símbolo a cada ΔT segundos. Na Figura 5.34(a) vemos um balde que contém três símbolos, com cinco pacotes aguardando para serem transmitidos. Para que um pacote seja transmitido, ele deve capturar e destruir um símbolo. Na Figura 5.34(b), vemos que três dos cinco pacotes percorreram todo o caminho, mas os outros dois estão emperrados, aguardando que dois outros símbolos sejam gerados.

O algoritmo de balde de símbolos possibilita um tipo de moldagem de tráfego diferente do algoritmo de balde furado. O algoritmo de balde furado não deixa que hosts inativos poupem permissões para enviar rajadas maiores posteriormente. O algoritmo de balde de símbolos permite economia, até o tamanho máximo do balde, *n*. Essa propriedade significa que rajadas de até *n* pacotes podem ser enviadas simultaneamente, permitindo um certo volume no fluxo de saída e possibilitando uma resposta mais rápida a rajadas de entrada repentinas.

Outra diferença entre os dois algoritmos é que o algoritmo de balde de símbolos joga símbolos fora (isto é, capacidade de transmissão) quando o balde enche,

mas nunca descarta pacotes. Em contrapartida, o algoritmo de balde furado descarta pacotes quando o balde fica cheio.

Aqui também é possível uma variante menos importante, em que cada símbolo representa o direito de enviar não um pacote, mas *k* bytes. Um pacote só poderá ser transmitido se houver símbolos suficientes disponíveis para abranger seu comprimento em bytes. Símbolos fracionários são mantidos para uso futuro.

Os algoritmos de balde furado e de balde de símbolos também podem ser usados para atenuar o tráfego entre roteadores, assim como podem ser usados para controlar a saída do host, como acontece em nossos exemplos. Entretanto, uma diferença óbvia é que um balde de símbolos que está controlando um host pode fazê-lo interromper a transmissão quando necessário. Ordenar que um roteador interrompa a transmissão quando ela já está em andamento pode resultar em perda de dados.

[arte: ver original p. 403]

[Dísticos]

[1](a) 25 MB/s durante 40 ms

0 Tempo (ms) 500

[2] (b)

2 MB/s durante 500 ms

0 Tempo (ms) 500

[3] (c) 25 MB/s durante 11 ms

2 MB/s durante 362 ms

0 Tempo (ms) 500

[4] (d) 25 MB/s durante 22 ms

2 MB/s durante 225 ms

0 Tempo (ms) 500

[5] (e) 25 MB/s durante 33 ms

2 MB/s durante 88 ms

0 Tempo (ms) 500

[6] (f) 10 MB/s durante 62 ms

2 MB/s durante 190 ms

0 Tempo (ms) 500

[F]Figura 5.33

[FL] (a) Entrada para um balde furado. (b) Saída de um balde furado. Saída de um balde de símbolos com capacidades de (c) 250 KB, (d) 500 KB e (e) 750 KB. (f) Saída de um balde de símbolos de 500 KB que alimenta um balde furado de 10 MB/s

A implementação do algoritmo de balde de símbolos é realizada apenas por meio de uma variável que conta símbolos. O contador é incrementado em uma unidade a cada intervalo de tempo ΔT e é decrementado de uma unidade sempre que um pacote é enviado. Quando o contador atinge zero, nenhum pacote pode ser enviado. Na variante de contagem de bytes, o contador é incrementado em k bytes a cada ΔT e decrementado de acordo com o comprimento de cada pacote enviado.

[arte: ver original p. 404]

[Dísticos]

[1] Computador host

Um símbolo é colocado no balde a cada ΔT

O balde armazena símbolos

Redes

(a)

[2]Computador host

Redes

(b)

[F]Figura 5.34

[FL] O algoritmo de balde de símbolos. (a) Antes. (b) Depois

Basicamente, o que o balde de símbolos faz é permitir rajadas, mas apenas até uma duração máxima controlada. Observe um exemplo na Figura 5.33(c). Nessa figura, temos um balde de símbolos com uma capacidade de 250 KB. Os símbolos chegam a uma taxa que permite a saída a 2 MB/s. Supondo-se que o balde de símbolos esteja cheio ao chegar a rajada de 1 MB, o escoamento do balde pode ser feito em uma velocidade plena de 25 MB/s durante cerca de 11 ms. Depois desse tempo, o balde de símbolos precisa diminuir para 2 MB/s até que a rajada de entrada tenha sido completamente enviada.

Calcular a duração da rajada à taxa máxima é um pouco complicado. O cálculo não é feito simplesmente dividindo-se 1 MB por 25 MB/s, porque chegam mais símbolos enquanto a rajada está sendo transmitida. Se chamarmos a duração da rajada de S segundos, a capacidade do balde de símbolos de C bytes, a taxa de chegada de símbolos de ρ bytes/s e a taxa de saída máxima de M bytes/s, vemos que uma rajada de saída contém no máximo $C + \rho S$ bytes. Também sabemos que o número de bytes em uma rajada à velocidade máxima de duração igual a S segundos é MS. Conseqüentemente, temos:

$$C + \rho S = MS$$

Podemos resolver essa equação para obter $S = C/(M - \rho)$. Para nossos parâmetros C = 250 KB, M = 25 MB/s e ρ = 2 MB/s, obtemos um tempo de rajada de cerca de 11 ms. As Figuras 5.33(d) e 5.33(e) mostram o balde de símbolos para capacidades de 500 KB e 750 KB, respectivamente.

Um problema potencial com o algoritmo de balde de símbolos é que ele permite que ocorram grandes rajadas novamente, ainda que o intervalo máximo entre

rajadas possa ser controlado por uma seleção cuidadosa de ρ e de *M*. Com freqüência, é interessante reduzir a taxa de pico, mas sem voltar ao valor baixo do balde furado original.

Uma forma de obter um tráfego mais suave é inserir um balde furado após o balde de símbolos. A taxa do balde furado deve ser maior que o valor ρ do balde de símbolos, mas deve ser inferior à taxa máxima da rede. A Figura 5.33(f) mostra a saída para um balde de símbolos de 500 KB seguido por um balde furado de 10 MB/s.

Policiar todos esses esquemas pode ser um pouco complicado. Em essência, a rede precisa simular o algoritmo e garantir que não estão sendo enviados mais pacotes ou bytes que o permitido. Não obstante, essas ferramentas fornecem meios para modelar o tráfego de rede em formas mais gerenciáveis, a fim de atender aos requisitos de qualidade de serviço.

[T4] Reserva de recursos

A capacidade de regular a forma do tráfego oferecido é um bom início para garantir a qualidade de serviço. Porém, o uso efetivo dessa informação significa implicitamente exigir que todos os pacotes de um fluxo sigam a mesma rota. Dispersar os pacotes pelos roteadores ao acaso torna difícil estabelecer qualquer garantia. Como conseqüência, algo semelhante a um circuito virtual tem de ser configurado desde a origem até o destino, e todos os pacotes que pertencem ao fluxo devem seguir essa rota.

Depois de termos uma rota específica para um fluxo, torna-se possível reservar recursos ao longo dessa rota, a fim de garantir que a capacidade necessária estará disponível. É possível reservar três diferentes tipos de recursos:

1. Largura de banda.
2. Espaço de buffer.

3. Ciclos da CPU.

O primeiro recurso, a largura de banda, é o mais óbvio. Se um fluxo exige 1 Mbps e a linha de partida tem uma capacidade de 2 Mbps, tentar orientar três fluxos por essa linha é uma estratégia que não vai funcionar. Desse modo, reservar largura de banda não significa sobrecarregar qualquer linha de saída.

Um segundo recurso freqüentemente escasso é o espaço em buffers. Quando um pacote chega, em geral ele é depositado na placa de interface de rede pelo próprio hardware. Em seguida, o software do roteador tem de copiá-lo para um buffer em RAM e enfileirar esse buffer para transmissão na linha de saída escolhida. Se nenhum buffer estiver disponível, o pacote terá de ser descartado, pois não há lugar para colocá-lo. Para se alcançar uma boa qualidade de serviço, alguns buffers podem ser reservados para um fluxo específico, de forma que o fluxo não tenha de competir pelos buffers com outros fluxos. Sempre haverá um buffer disponível quando o fluxo precisar dele, até algum número máximo.

Finalmente, os ciclos da CPU também constituem um recurso escasso. O processamento de um pacote exige tempo da CPU do roteador, e assim o roteador só pode processar um certo número de pacotes por segundo. É necessário ter certeza de que a CPU não está sobrecarregada, a fim de assegurar o processamento oportuno de cada pacote.

À primeira vista pode parecer que, se a demora é de, digamos, 1 µs para processar um pacote, um roteador pode processar 1 milhão de pacotes por segundo. Essa observação não é verdadeira, porque sempre haverá períodos ociosos devido a flutuações estatísticas na carga. Se a CPU precisar de cada ciclo para realizar seu trabalho, até mesmo a perda de alguns ciclos em conseqüência de períodos ociosos ocasionais cria um acúmulo de serviço do qual ela nunca conseguirá se livrar.

No entanto, até mesmo com uma carga ligeiramente abaixo da capacidade

teórica, as filas podem aumentar e os retardos podem ocorrer. Considere uma situação na qual os pacotes chegam ao acaso, com uma taxa média de chegadas igual a λ pacotes/s. O tempo de CPU exigido por cada um deles também é aleatório, sendo a capacidade média de processamento igual a μ pacotes/s. Sob a hipótese de que tanto a distribuição de chegadas quanto a distribuição de serviços sejam distribuições de Poisson, podemos provar, usando a teoria do enfileiramento, que o retardo médio *T* experimentado por um pacote é:

[Inserir equação do O.A. p. 406]

onde $\rho = \lambda/\mu$ é a utilização da CPU. O primeiro fator, $1/\mu$, é o valor que o tempo de serviço teria na ausência de competição. O segundo fator é a redução da velocidade devido à competição com outros fluxos. Por exemplo, se $\lambda = 950.000$ pacotes/s e $\mu = 1.000.000$ pacotes/s, então $\rho = 0,95$, e o retardo médio experimentado por cada pacote será igual a 20 µs, em vez de 1 µs. Esse tempo leva em conta o tempo de enfileiramento e o tempo de serviço, como podemos observar quando a carga é muito baixa ([ver símbolo]). Se houver, digamos, 30 roteadores ao longo da rota do fluxo, o retardo de enfileiramento responderá sozinho pelo retardo de 600 µs.

[T4] Controle de admissão

Agora chegamos ao ponto em que o tráfego de entrada de algum fluxo é bem modelado e pode potencialmente seguir uma única rota na qual a capacidade pode ser reservada com antecedência nos roteadores ao longo do caminho. Quando tal fluxo é oferecido a um roteador, ele tem de decidir, com base em sua capacidade e na quantidade de compromissos que já assumiu para outros fluxos, se deve admitir ou rejeitar o fluxo.

A decisão de aceitar ou rejeitar um fluxo não é uma simples questão de comparar os recursos (largura de banda, buffers, ciclos) solicitados pelo fluxo com a

capacidade em excesso do roteador nessas três dimensões. O problema é um pouco mais complicado. Para começar, embora algumas aplicações possam conhecer seus requisitos de largura de banda, poucas sabem algo sobre buffers ou ciclos da CPU; então, no mínimo, é necessário encontrar uma forma diferente de descrever fluxos. Em seguida, algumas aplicações são muito mais tolerantes à perda ocasional de um prazo final que outras. Finalmente, algumas aplicações podem estar dispostas a pechinchar sobre os parâmetros de fluxo, e outras não. Por exemplo, um visualizador de filmes que normalmente funciona a 30 quadros/s pode estar disposto a reduzir sua velocidade para 25 quadros/s, se não houver largura de banda suficiente para admitir 30 quadros/s. De modo semelhante, o número de pixels por quadro, a largura de banda de áudio e outras propriedades podem ser ajustáveis.

Como muitas partes talvez estejam envolvidas na negociação de fluxo (o transmissor, o receptor e todos os roteadores ao longo do caminho entre eles), os fluxos devem ser descritos com precisão em termos de parâmetros específicos que podem ser negociados. Um conjunto desses parâmetros é chamado **especificação de fluxo.** Em geral, o transmissor (por exemplo, o servidor de vídeo) produz uma especificação de fluxo propondo os parâmetros que ele gostaria de usar. À medida que a especificação se propaga ao longo da rota, cada roteador a examina e modifica os parâmetros conforme necessário. As modificações só podem reduzir o fluxo, não aumentá-lo (por exemplo, uma taxa de dados mais baixa, e não mais alta). Quando ela chega à outra extremidade, os parâmetros podem ser estabelecidos.

Como exemplo do que pode haver em uma especificação de fluxo, considere o exemplo da Figura 5.35, baseada nas RFCs 2210 e 2211. Ela tem cinco parâmetros; o primeiro, a *taxa de balde de símbolos*, é o número de bytes por segundo que são inseridos no balde. Essa é a taxa máxima sustentada em que o

transmissor pode transmitir, calculada com uma média sobre um longo intervalo de tempo.

[arte: ver original p. 407]

[T]Tabela

Parâmetro	**Unidade**
Taxa de balde de símbolos	Bytes/s
Tamanho do balde de símbolos	Bytes
Taxa de dados de pico	Bytes/s
Tamanho mínimo do pacote	Bytes
Tamanho máximo do pacote	Bytes

[F]Figura 5.35

[FL] Um exemplo de especificação de fluxo

O segundo parâmetro é o tamanho do balde em bytes. Se, por exemplo, a *taxa de balde de símbolos* é 1 Mbps e o *tamanho de balde de símbolos* é 500 KB, o balde pode se encher continuamente durante 4 segundos antes de se encher por completo (na ausência de quaisquer transmissões). Quaisquer símbolos enviados depois disso são perdidos.

O terceiro parâmetro, a *taxa de dados de pico*, é a taxa máxima de transmissão tolerada, mesmo durante breves intervalos de tempo. O transmissor nunca deve exceder essa taxa.

Os dois últimos parâmetros especificam os tamanhos mínimo e máximo do pacote, incluindo os cabeçalhos da camada de transporte da camada de rede (por exemplo, TCP e IP). O tamanho mínimo é importante, porque o processamento de cada pacote demora algum tempo fixo, independente de quanto ele seja curto.

Um roteador pode estar preparado para manipular 10.000 pacotes/s de 1 KB cada, mas não estar preparado para tratar 100.000 pacotes/s de 50 bytes cada,

embora isso represente uma taxa de dados mais baixa. O tamanho máximo do pacote é importante, devido a limitações internas da rede que não podem ser excedidas. Por exemplo, se parte do caminho passar por uma rede Ethernet, o tamanho máximo do pacote será restrito a não mais de 1500 bytes, não importando que tamanho o restante da rede possa manipular.

Uma pergunta interessante é como um roteador transforma uma especificação de fluxo em um conjunto de reservas de recursos específicos. Esse mapeamento é específico da implementação e não é padronizado. Suponha que um roteador possa processar 100.000 pacotes/s. Se for oferecido a ele um fluxo de 1 MB/s com tamanhos mínimo e máximo de pacote iguais a 512 bytes, o roteador poderá calcular que deve receber 2048 pacotes/s desse fluxo. Nesse caso, ele tem de reservar 2% de sua CPU para o fluxo, de preferência mais, a fim de evitar longos retardos de enfileiramento. Se a política de um roteador for a de nunca alocar mais de 50% de sua CPU (o que implica um fator de retardo igual a 2), e ela já estiver com 49% de sua capacidade ocupados, esse fluxo terá de ser rejeitado. Cálculos semelhantes serão necessários para os outros recursos.

Quanto mais restrita a especificação de fluxo, mais útil ela será para os roteadores. Se uma especificação de fluxo declarar que precisa de uma *taxa de balde de símbolos* igual a 5 MB/s, mas os pacotes puderem variar de 50 bytes a 1500 bytes, então a taxa de pacotes irá variar de cerca de 3500 pacotes/s para 105.000 pacotes/s. O roteador poderá entrar em pânico com esse último número e rejeitar o fluxo, enquanto um tamanho mínimo de pacote igual a 1000 bytes poderia fazer o fluxo de 5 MB/s ter sido aceito.

[T4] Roteamento proporcional

A maioria de algoritmos de roteamento tenta encontrar o melhor caminho para cada destino e enviar todo tráfego para esse destino pelo melhor caminho. Uma

abordagem diferente, que foi proposta para oferecer uma qualidade de serviço mais alta, é dividir o tráfego correspondente a cada destino entre vários caminhos. Tendo em vista que os roteadores em geral não têm uma visão completa do tráfego de toda a rede, a única maneira viável de dividir o tráfego por várias rotas é usar informações disponíveis no local. Um método simples é dividir o tráfego igualmente ou torná-lo proporcional à capacidade dos enlaces de saída. No entanto, também estão disponíveis algoritmos mais sofisticados (Nelakuditi e Zhang, 2002).

[T4] Programação de pacotes

Se um roteador estiver tratando vários fluxos, haverá o perigo de um fluxo receber muito mais do que permite sua capacidade e deixar todos os outros fluxos à mingua. O processamento de pacotes na ordem de sua chegada significa que um transmissor agressivo pode capturar a maior parte da capacidade dos roteadores por onde passam seus pacotes, reduzindo a qualidade de serviço para outros. Para contrariar tais tentativas, foram criados vários algoritmos para programação de pacotes (Bhatti e Crowcroft, 2000).

Um dos primeiros foi o algoritmo de **enfileiramento justo** (Nagle, 1987). A essência do algoritmo é que os roteadores têm filas separadas para cada linha de saída, uma para cada fluxo. Quando uma linha fica ociosa, o roteador varre as filas em rodízio, tomando o primeiro pacote da fila seguinte. Desse modo, com *n* hosts competindo por uma dada linha de saída, cada host chega a enviar um dentre cada *n* pacotes. A transmissão de mais pacotes não irá melhorar essa fração.

Apesar de ser um bom início, o algoritmo tem um problema: ele fornece mais largura de banda para hosts que utilizam pacotes grandes do que para hosts que utilizam pacotes pequenos. Demers *et al.* (1990) sugeriram um aperfeiçoamento

no qual o rodízio é feito de forma a simular um rodízio byte a byte, em vez de um rodízio pacote a pacote. Na verdade, ele percorre as filas repetidamente, byte por byte, até encontrar o pulso em que cada pacote terminará. Em seguida, os pacotes são classificados em ordem de término e enviados nessa ordem. O algoritmo é ilustrado na Figura 5.36.

[arte: ver original p. 409]

[Dísticos]

[1] Pacote

[2] Tempo de término

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.36

[FL] (a) Um roteador com cinco pacotes enfileirados para a linha *O*. (b) Tempos de término dos cinco pacotes

Na Figura 5.36(a), vemos pacotes de 2 a 6 bytes. No pulso do clock (virtual) 1, o primeiro byte do pacote na linha *A* é enviado. Depois, é transmitido o primeiro byte do pacote da linha *B* e assim por diante. O primeiro pacote a terminar é *C*, depois de oito pulsos. A ordem classificada é mostrada na Figura 5.36(b). Na ausência de novas chegadas, os pacotes serão enviados na ordem listada, de *C* a *A*.

Um problema com esse algoritmo é que ele dá a todos os hosts a mesma prioridade. Em muitas situações, talvez seja interessante dar a servidores de vídeo maior largura de banda que a servidores de arquivos comuns, de forma que eles possam receber dois ou mais bytes por pulso. Esse algoritmo modificado é chamado algoritmo de **enfileiramento justo ponderado** e é amplamente utilizado.

Às vezes, o peso é igual ao número de fluxos que saem de uma máquina; portanto, cada processo recebe uma largura de banda equivalente. Uma implementação eficiente do algoritmo é analisada em (Shreedhar e Varghese, 1995). Cada vez mais, o encaminhamento real de pacotes por um roteador ou switch está sendo feito em hardware (Elhanany *et al.*, 2001).

[T3] 5.4.3 Serviços integrados

Entre 1995 e 1997, a IETF dedicou um grande esforço à criação de uma arquitetura para multimídia de fluxo. Esse trabalho resultou em mais de duas dezenas de RFCs, começando com as RFCs 2205 a 2210. O nome genérico desse trabalho é **algoritmos baseados no fluxo** ou **serviços integrados.** Ele teve como objetivo as aplicações de unidifusão e multidifusão. Um exemplo do primeiro tipo de aplicação é um único usuário que recebe um fluxo de um videoclipe transmitido por um site de notícias. Um exemplo do outro tipo de aplicação é um conjunto de estações de televisão digital que transmitem seus programas sob a forma de fluxos de pacotes IP para muitos receptores situados em diversos locais. Vamos nos conectar a seguir na multidifusão, pois a unidifusão é um caso especial de multidifusão.

Em muitas aplicações de multidifusão, os grupos podem alterar seus membros dinamicamente; por exemplo, quando as pessoas entram em uma videoconferência ou se entediam e passam para uma novela ou para o canal de esportes. Nessas condições, a estratégia de fazer com que os transmissores reservem largura de banda com antecedência não funciona muito bem, pois ela exigiria que cada transmissor rastreasse todas as entradas e saídas de sua audiência. No caso de um sistema projetado para transmitir imagens de televisão a cabo, com milhões de assinantes, esse esquema não funcionaria de forma alguma.

[T4] RSVP — Resource reSerVation Protocol

O principal protocolo da IETF para a arquitetura de serviços integrados é o **RSVP**, descrito na RFC 2205 e em outras. Esse protocolo é empregado para fazer as reservas; outros protocolos são usados para transmitir os dados. O RSVP permite que vários transmissores enviem os dados para vários grupos de receptores, torna possível receptores individuais mudarem livremente de canais e otimiza o uso da largura de banda ao mesmo tempo que elimina o congestionamento.

Em sua forma mais simples, o protocolo utiliza roteamento por multidifusão com árvores de amplitude, como discutimos anteriormente. Cada grupo recebe um endereço de grupo. Para transmitir dados a um grupo, um transmissor coloca o endereço desse grupo em seus pacotes. Em seguida, o algoritmo de roteamento por multidifusão padrão constrói uma árvore de amplitude que cobre todos os membros. O algoritmo de roteamento não faz parte do RSVP. A única diferença em relação à multidifusão normal são algumas informações extras transmitidas periodicamente ao grupo por multidifusão, a fim de informar aos roteadores ao longo da árvore que devem manter certas estruturas de dados em suas respectivas memórias.

Como exemplo, considere a rede da Figura 5.37(a). Os hosts 1 e 2 são transmissores de multidifusão, e os hosts 3, 4 e 5 são receptores de multidifusão. Nesse exemplo, os transmissores e os receptores estão separados mas, em geral, os dois conjuntos podem se sobrepor. As árvores de multidifusão para os hosts 1 e 2 são mostradas na Figura 5.37(b) e na Figura 5.37(c), respectivamente.

Para obter uma melhor recepção e eliminar o congestionamento, qualquer um dos receptores de um grupo pode enviar uma mensagem de reserva pela árvore para o transmissor. A mensagem é propagada com a utilização do algoritmo de

encaminhamento pelo caminho inverso, discutido antes. Em cada hop, o roteador detecta a reserva e guarda a largura de banda necessária. Se a largura de banda disponível não for suficiente, ele reporta a falha. No momento em que a mensagem retornar à origem, a largura de banda já terá sido reservada ao longo de todo o caminho entre o transmissor e o receptor, fazendo a solicitação de reserva ao longo da árvore de amplitude.

Um exemplo desse processo de reserva é mostrado na Figura 5.38(a). Aqui, o host 3 solicitou um canal ao host 1. Uma vez estabelecido o canal, os pacotes podem fluir do host 1 até o host 3, sem congestionamento. Agora, considere o que acontecerá se, em seguida, o host 3 reservar um canal para o outro transmissor, o host 2, de forma que o usuário possa assistir a dois programas de televisão ao mesmo tempo. Um segundo caminho será reservado, como ilustra a Figura 5.38(b). Observe que são necessários dois canais distintos entre o host 3 e o roteador *E*, pois dois fluxos independentes estão sendo transmitidos.

[arte: ver original p. 411]

[Dísticos]

[1]Transmissores

[2]Receptores

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.37

[FL] (a) Uma rede. (b) A árvore de amplitude de multidifusão para o host 1. (c) A árvore de amplitude de multidifusão para o host 2

Por fim, na Figura 5.38(c), o host 5 decide assistir ao programa que está sendo transmitido pelo host 1 e também faz uma reserva. Primeiro, é reservada uma

largura de banda dedicada até o roteador *H*. Entretanto, esse roteador percebe que já está sendo alimentado pelo host 1; sendo assim, como a largura de banda necessária já foi reservada, ele não precisa reservar mais nada. Observe que os hosts 3 e 5 poderiam ter solicitado diferentes volumes de largura de banda (por exemplo, o host 3 tem um aparelho de televisão em preto e branco, e assim não deseja as informações sobre cores); portanto, a capacidade reservada deve ser grande o suficiente para satisfazer ao receptor mais voraz.

Ao fazer uma reserva, um receptor pode (opcionalmente) especificar uma ou mais origens a partir das quais deseja receber informações. Ele também pode especificar se essas opções serão fixas durante o período de reserva, ou se o receptor deseja manter em aberto a opção de alterar as origens mais tarde. Os roteadores utilizam essas informações para otimizar o planejamento da largura de banda. Em particular, dois receptores só são configurados para compartilhar um caminho se ambos concordarem em não alterar as origens posteriormente.

O motivo para essa estratégia no caso totalmente dinâmico é que a largura de banda reservada é desacoplada da opção de origem. Quando reserva a largura de banda, um receptor pode alternar para outra origem e manter a parte do caminho existente que for válida para a nova origem. Por exemplo, se o host 2 estiver transmitindo diversos fluxos de vídeo, o host 3 poderá alternar entre eles quando quiser sem alterar sua reserva, pois os roteadores não se importam com o programa a que o receptor está assistindo.

[arte: ver original p. 412]

[Dísticos]

[1] Largura de banda reservada para a origem 1

[2] Largura de banda reservada para a origem 2

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem

seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.38

[FL] (a) O host 3 solicita um canal ao host 1. (b) Em seguida, o host 3 solicita um segundo canal, agora ao host 2. (c) O host 5 solicita um canal ao host 1

[T3] 5.4.4 Serviços diferenciados

Os algoritmos baseados no fluxo têm potencial para oferecer boa qualidade de serviço a um ou mais fluxos, porque eles reservam quaisquer recursos necessários ao longo da rota. Porém, eles também têm a desvantagem de exigirem uma configuração antecipada para estabelecer cada fluxo, algo que não se ajusta bem quando existem milhares ou milhões de fluxos. Além disso, eles mantêm o estado interno por fluxo nos roteadores, tornando-os vulneráveis a quedas de roteadores. Por fim, as mudanças exigidas no código do roteador são substanciais e envolvem trocas complexas de roteador para roteador, a fim de configurar os fluxos. Em conseqüência disso, ainda existem poucas implementações de RSVP ou de algo semelhante.

Por essas razões, a IETF também criou uma abordagem mais simples para oferecer qualidade de serviço, uma estratégia que pode ser implementada em grande parte no local em cada roteador, sem configuração antecipada e sem ter de envolver todo o caminho. Essa abordagem é conhecida como qualidade de serviço **baseada na classe** (em vez de ser baseada no fluxo). A IETF padronizou uma arquitetura para ela, chamada arquitetura de **serviços diferenciados**, descrita nas RFCs 2474, 2475 e várias outras. Vamos descrevê-la agora.

Os serviços diferenciados (DS — differentiated services) podem ser oferecidos por um conjunto de roteadores que formam um domínio administrativo (por exemplo, um ISP ou uma empresa de telecomunicações). A administração define um conjunto de classes de serviço com regras de encaminhamento correspondentes.

Se um cliente fizer a assinatura para DS, os pacotes do cliente que entrarem no domínio poderão incluir um campo *Tipo de serviço*, sendo fornecido um serviço melhor a algumas classes (por exemplo, um serviço especial) que a outras. O tráfego dentro de uma classe talvez tenha de obedecer a alguma forma específica, como a de um balde furado com alguma taxa de escoamento especificada. Uma operadora com um certo tino comercial poderia cobrar uma tarifa extra por cada pacote especial transportado, ou poderia permitir até *N* pacotes especiais por mês a uma taxa mensal adicional fixa. Observe que esse esquema não exige nenhuma configuração antecipada, nenhuma reserva de recursos e nenhuma negociação demorada de fim a fim para cada fluxo, como ocorre no caso dos serviços integrados. Isso torna relativamente fácil implementar os serviços diferenciados.

O serviço baseado na classe também ocorre em outros campos. Por exemplo, as empresas de entrega de pacotes freqüentemente oferecem serviço noturno, em dois dias e em três dias. As empresas aéreas oferecem serviço de primeira classe, da classe executiva e da classe econômica. Os trens interurbanos muitas vezes têm várias classes de serviço. Até mesmo o metrô de Paris tem duas classes de serviço. No caso dos pacotes, as classes de serviço podem diferir em termos de retardo, flutuação e probabilidade de os pacotes serem descartados na eventualidade de ocorrer congestionamento, entre outras possibilidades (mas talvez não de quadros Ethernet, mais espaçosos).

Para tornar mais clara a diferença entre a qualidade de serviço baseada no fluxo e a qualidade de serviço baseada na classe, vamos considerar o exemplo da telefonia na Internet. Com um esquema baseado no fluxo, cada chamada telefônica obtém seus próprios recursos e suas garantias. Com um esquema baseado na classe, todas as chamadas telefônicas juntas obtêm os recursos reservados para a telefonia da classe. Esses recursos não podem ser tirados pelos

pacotes da classe de transferência de arquivos ou de outras classes, mas nenhuma chamada telefônica recebe quaisquer recursos privados reservados apenas para ela.

[T4] Encaminhamento expedido

A escolha de classes de serviço cabe a cada operadora mas, como os pacotes com freqüência são encaminhados entre sub-redes pertencentes a diferentes operadoras, a IETF está trabalhando na definição de classes de serviço independentes da rede. A mais simples dessas classes é a de **encaminhamento expedido**; portanto, vamos começar por ela. Essa classe é descrita na RFC 3246.

A idéia que rege o encaminhamento expedido é muito simples. Há duas classes de serviço disponíveis: regular e expedido. A vasta maioria do tráfego deve ser regular, mas uma pequena fração dos pacotes é expedida. Os pacotes expedidos devem ser capazes de transitar pela sub-rede como se nenhum outro pacote estivesse presente. Uma representação simbólica desse sistema de "dois tubos" é dada na Figura 5.39. Observe que ainda existe apenas uma linha física. Os dois canais lógicos mostrados na figura representam um modo de reservar largura de banda, e não uma segunda linha física.

Um modo de implementar essa estratégia é programar os roteadores para terem duas filas de saída correspondentes a cada linha de saída, uma para pacotes expedidos e outra para pacotes regulares. Quando um pacote chega, ele enfileirado de acordo com seu tipo. A programação de pacotes deve usar algo semelhante ao enfileiramento justo ponderado. Por exemplo, se 10% do tráfego for expedido e 90% for regular, 20% da largura de banda poderá ser dedicada ao tráfego expedido, e o restante ao tráfego regular. Isso daria ao tráfego expedido o dobro da largura de banda de que ele necessitasse, com a finalidade de proporcionar baixo retardo para esse tipo de tráfego. Essa alocação pode ser

alcançada transmitindo um pacote expedido para cada quatro pacotes regulares (supondo-se que a distribuição de tamanhos para ambas as classes seja semelhante). Desse modo, espera-se que os pacotes expedidos encontrem uma sub-rede não carregada, mesmo quando houver de fato uma carga pesada.

[arte: ver original p. 414a]

[Dísticos]

[1] Pacotes expedidos

[2] Pacotes regulares

[F]Figura 5.39

[FL] Pacotes expedidos encontram uma rede livre de tráfego

[T4] Encaminhamento garantido

Um esquema um pouco mais elaborado para gerenciar as classes de serviço é chamado **encaminhamento garantido**. Ele é descrito na RFC 2597 e especifica que haverá quatro classes de prioridade, e cada classe terá seus próprios recursos. Além disso, ele define três probabilidades de descarte de pacotes que estejam sofrendo congestionamento: baixo, médio e alto. Considerados em conjunto, esses dois fatores definem 12 classes de serviço.

A Figura 5.40 mostra uma forma possível de processar pacotes no esquema de encaminhamento garantido. A etapa 1 consiste em classificar os pacotes em uma das quatro classes de prioridade. Essa etapa poderia ser feita no host transmissor (como mostra a figura) ou no roteador de ingresso (o primeiro). A vantagem de fazer a classificação no host transmissor é que nesse local há mais informações disponíveis sobre quais pacotes pertencem a quais fluxos.

[arte: ver original p. 414b]

[Dísticos]

[1] Pacotes Classificador Marcador Modelador/Regulador

[2] Quatro classes de prioridade

[3] Roteador de ingresso

[4] Linha de saída

[5] Classe

[6] Pacote enfileirado

[F]Figura 5.40

[FL] Uma implementação possível do fluxo de dados para encaminhamento garantido

A etapa 2 é a marcação dos pacotes de acordo com sua classe. É necessário um campo de cabeçalho para esse propósito. Felizmente, há um campo de 8 bits *Type of service* disponível no cabeçalho IP, como veremos em breve. A RFC 2597 especifica que seis desses bits devem ser usados para a classe de serviço, deixando espaço de codificação para classes de serviço históricas e futuras.

A etapa 3 consiste em fazer os pacotes passarem por um filtro modelador/regulador que pode retardar ou descartar alguns deles para modelar os quatro fluxos em formas aceitáveis; por exemplo, usando o balde furado ou o balde de símbolos. Se houver muitos pacotes, alguns deles talvez sejam descartados aqui, pela categoria de descarte. Também são possíveis esquemas mais elaborados, envolvendo medição ou feedback.

Nesse exemplo, essas três etapas são executadas no host de transmissão, e assim o fluxo de saída é agora entregue ao roteador de ingresso. Vale a pena notar que essas etapas podem ser executadas por software em rede especial, ou até mesmo pelo sistema operacional, a fim de evitar a necessidade de trocas entre aplicações existentes.

[T3] 5.4.5 Troca de rótulos e MPLS

Enquanto a IETF estava desenvolvendo serviços integrados e serviços diferenciados, vários fabricantes de roteadores estavam trabalhando em métodos de encaminhamento melhores. Esse trabalho se concentrou na inclusão de um rótulo (label) no início de cada pacote e na execução do roteamento baseado no rótulo, e não no endereço de destino. Fazer do rótulo um índice para uma tabela interna torna a localização da linha de saída correta apenas uma questão de pesquisa em tabela. Utilizando-se essa técnica, o roteamento pode ser feito com muita rapidez, e quaisquer recursos necessários podem ser reservados ao longo do caminho.

É claro que a identificação dos fluxos dessa maneira chega perigosamente perto dos circuitos virtuais. As redes X.25, ATM, frame relay e todas as outras redes com uma sub-rede de circuito virtual também incluem um rótulo (isto é, um identificador de circuito virtual) em cada pacote, realizam a pesquisa de rótulos em uma tabela e efetuam o roteamento com base na entrada da tabela. A despeito do fato de muitas pessoas na comunidade da Internet terem uma intensa antipatia pelas redes orientadas a conexões, a idéia parece ser recorrente, e dessa vez para permitir o roteamento rápido e oferecer qualidade de serviço. Porém, existem diferenças essenciais entre o modo como a Internet trata a construção de rotas e o modo como a construção de rotas é tratada nas redes orientadas a conexões; portanto, a técnica certamente não é a comutação de circuitos tradicional.

Essa "nova" idéia de comutação passa por vários nomes (patenteados), inclusive **comutação de rótulos** e **comutação de tags**. Eventualmente, a IETF começou a padronizar a idéia sob o nome **MPLS (MultiProtocol Label Switching — comutação de rótulos multiprotocolo)**. Vamos denominá-la MPLS no texto a seguir. Ela é descrita na RFC 3031 e em muitas outras RFCs.

A propósito, algumas pessoas fazem distinção entre *roteamento* e *comutação*.

Roteamento é o processo que consiste em procurar um endereço de destino em uma tabela, a fim de descobrir para onde enviar um pacote. Em contraste, a comutação utiliza um rótulo tirado do pacote como um índice para uma tabela de encaminhamento. Porém, essas definições estão longe de ser universais.

O primeiro problema é onde pôr o rótulo. Tendo em vista que os pacotes IP não foram projetados para circuitos virtuais, não existe nenhum campo disponível para números de circuitos virtuais dentro do cabeçalho IP. Por essa razão, surgiu a necessidade de adicionar um novo cabeçalho MPLS antes do cabeçalho IP. Em uma linha de roteador para roteador e usando-se o PPP como protocolo de enquadramento, o formato do quadro, incluindo os cabeçalhos PPP, MPLS, IP e TCP, é semelhante ao da Figura 5.41. De certo modo, pode-se considerar que o MPLS é a camada 2,5.

[arte: ver original p. 416]

[Dísticos]

[1]Cabeçalhos

PPP MPLS IP TCP Dados do usuário CRC

[2]Bits 20 3 1 8

Label QoS S TTL

[F]Figura 5.41

[FL] Transmitindo um segmento TCP com a utilização de IP, MPLS e PPP

O cabeçalho MPLS genérico tem quatro campos, sendo o mais importante o campo *Label*, que contém o índice. O campo *QoS* indica a classe de serviço. O campo *S* se relaciona ao empilhamento de vários rótulos em redes hierárquicas (que descreveremos a seguir). Se ele alcançar 0, o pacote será descartado. Esse recurso impede a entrada em loop infinito em caso de instabilidade de roteamento.

Como os cabeçalhos MPLS não fazem parte do pacote da camada de rede ou do quadro da camada de enlace de dados, considera-se o MPLS em grande parte independente de ambas as camadas. Entre outras coisas, essa propriedade significa que é possível construir switches MPLS que podem encaminhar tanto pacotes IP quanto células ATM, dependendo do tipo de objeto que surgir. Essa característica explica a palavra "multiprotocolo" no nome MPLS.

Quando um pacote (ou uma célula) aperfeiçoado pelo MPLS chega a um roteador capaz de reconhecer o MPLS, o rótulo é usado como um índice para uma tabela, a fim de determinar a linha de saída que deve ser usada, e também qual o novo rótulo. Essa troca de rótulos é utilizada em todas as sub-redes de circuitos virtuais, porque os rótulos têm significado apenas local, e dois roteadores diferentes podem alimentar pacotes não relacionados com o mesmo rótulo para outro roteador, de forma que a transmissão seja feita na mesma linha de saída. Para que os pacotes possam ser reconhecidos na outra extremidade, os rótulos têm de ser remapeados a cada hop. Vimos esse mecanismo em ação na Figura 5.3. O MPLS usa a mesma técnica.

Uma diferença em relação aos circuitos virtuais tradicionais é o nível de agregação. Sem dúvida, é possível cada fluxo ter seu próprio conjunto de rótulos na sub-rede. Porém, é mais comum os roteadores agruparem vários fluxos que terminam em um certo roteador ou LAN e usarem um único rótulo para eles. Dizemos que os fluxos agrupados sob um único rótulo pertencem à mesma **FEC (Forwarding Equivalence Class — classe de equivalência de encaminhamento)**. Essa classe abrange não apenas os lugares para onde os pacotes estão indo, mas também sua classe de serviço (no sentido de serviços diferenciados), porque todos os seus pacotes são tratados do mesmo modo para fins de encaminhamento.

Com o roteamento tradicional de circuito virtual, não é possível agrupar vários

caminhos distintos com pontos extremos diferentes no mesmo identificador de circuito virtual, porque não haveria como distingui-los no destino final. No caso do MPLS, os pacotes ainda contêm seu endereço de destino final, além do rótulo, e assim no final da rota identificada, o cabeçalho do rótulo poderá ser removido e o encaminhamento poderá continuar da maneira habitual, com a utilização de um endereço de destino da camada de rede.

Uma diferença importante entre o MPLS e os projetos convencionais de circuitos virtuais é o modo como a tabela de encaminhamento é construída. Nas redes de circuitos virtuais tradicionais, quando um usuário quer estabelecer uma conexão, é lançado na rede um pacote de configuração para criar o caminho e gerar as entradas da tabela de encaminhamento. O MPLS não funciona dessa maneira, porque não existe uma fase de configuração para cada conexão (pois isso impediria a utilização de um software da Internet, existente há muito tempo).

Em vez disso, há duas maneiras de criar as entradas da tabela de encaminhamento. Na abordagem **orientada para dados**, quando um pacote chega, o primeiro roteador que ele acessa entra em contato com o roteador situado mais abaixo por onde o pacote deve passar e pede que ele gere um rótulo para o fluxo. Esse método é aplicado recursivamente. Na verdade, esse é um exemplo de criação de circuito virtual por demanda.

Os protocolos que realizam essa dispersão têm o cuidado de evitar loops. Com freqüência, eles utilizam uma técnica chamada **threads coloridos**. A propagação de uma FEC no sentido inverso pode ser comparada à ação de puxar um thread (ou fio) colorido exclusivamente de volta para a sub-rede. Se um roteador detectar uma cor que já tem, ele saberá que existe um loop e executará uma ação corretiva. A abordagem orientada para os dados é usada principalmente em redes nas quais o transporte subjacente é do tipo ATM (como ocorre em grande parte do sistema telefônico).

A outra abordagem, usada em redes não baseadas no ATM, é a abordagem **orientada por controle**, que apresenta diversas variações. Uma delas funciona da maneira explicada a seguir. Quando é inicializado, um roteador verifica para quais rotas ele é o destino final (por exemplo, quais hosts estão em sua LAN). Em seguida, ele cria uma ou mais FECs para elas, aloca um rótulo para cada uma e repassa os rótulos a seus vizinhos. Por sua vez, eles inserem os rótulos em suas tabelas de encaminhamento e enviam novos rótulos a seus vizinhos, até todos os roteadores se apossarem do caminho. Os recursos também podem ser reservados à medida que o caminho é construído, a fim de garantir uma qualidade de serviço apropriada.

O MPLS pode operar em vários níveis ao mesmo tempo. No nível mais alto, cada concessionária pode ser considerada uma espécie de metarroteador, existindo um caminho pelo metarroteadores, desde a origem até destino. Esse caminho pode usar o MPLS. No entanto, dentro da rede de cada concessionária, o MPLS também pode ser usado, o que nos leva a um segundo nível de identificação. De fato, um pacote pode transportar uma pilha inteira de rótulos. O bit *S* da Figura 5.41 permite a um roteador remover um rótulo para saber se existem rótulos adicionais. Ele é definido como 1 para indicar o rótulo inferior e como 0 para todos os outros rótulos. Na prática, esse recurso é usado principalmente para implementar redes privadas virtuais e túneis recursivos.

Embora as idéias básicas sob o MPLS sejam simples, os detalhes são extremamente complicados, com muitas variações e otimizações; assim, não continuaremos a analisar esse tópico. Para obter mais informações, consulte (Davie e Rekhter, 2000; Lin *et al.*, 2002; Pepelnjak e Guichard, 2001; e Wang, 2001).

[T2] 5.5 Interligação de redes

Até agora, supomos implicitamente que havia uma única rede homogênea, com cada máquina usando o mesmo protocolo em cada camada. Porém, essa suposição é muito otimista. Existem muitas redes diferentes, incluindo LANs, MANs e WANs. Diversos protocolos estão sendo bastante utilizados em cada camada. Nas seções a seguir, examinaremos cuidadosamente as questões que surgem quando duas ou mais redes são interconectadas para formar uma **inter-rede.**

Existe considerável controvérsia em relação a atual abundância de tipos de redes. Não sabemos se essa é uma condição temporária que passará tão logo alguém perceba o quanto a rede [preencha com a sua rede favorita] é maravilhosa, ou se essa é uma característica inevitável, mas permanente, que veio para ficar. Ter diferentes redes invariavelmente também significa ter diferentes protocolos.

Acreditamos que sempre haverá uma variedade de redes com características (e protocolos) distintos por vários motivos. Em primeiro lugar, a base instalada dos diferentes tipos de redes é grande. Quase todos os computadores pessoais utilizam o TCP/IP. Muitas empresas de grande porte têm mainframes que utilizam a SNA da IBM. Um número significativo de companhias telefônicas opera redes ATM. Algumas LANs de computadores pessoais ainda utilizam o Novell NCP/IPX ou o AppleTalk. Por fim, as redes sem fios constituem uma área nova com uma variedade de protocolos. Essa tendência continuará ainda durante muitos anos devido a problemas de compatibilidade com tecnologias antigas, a novas tecnologias e ao fato de que nem todos os fabricantes percebem que ela é de seu interesse, pois seus clientes são capazes de migrar facilmente para o sistema de outro fornecedor.

Em segundo lugar, como os computadores e as redes estão se tornando mais econômicos, as decisões passam a ser tomadas em níveis mais baixos na hierarquia das organizações. Muitas empresas têm uma política efetiva de que os

custos superiores a um milhão de dólares devem ser aprovados pela gerência de nível hierárquico mais alto, enquanto os custos superiores a 100.000 dólares devem ser aprovados pela gerência de nível médio. No entanto, as aquisições inferiores a 100.000 dólares podem ser feitas por chefes de departamentos, sem qualquer aprovação superior. Essa situação pode levar o departamento de engenharia a instalar estações de trabalho UNIX executando o TCP/IP e o departamento de marketing a instalar computadores Macintosh com AppleTalk.
Em terceiro lugar, os diferentes tipos de redes (por exemplo, ATM e sem fio) têm tecnologias radicalmente distintas; por isso, não será surpresa que, à medida que ocorrerem novos desenvolvimentos de hardware, também sejam criados novos softwares correspondentes. Por exemplo, o padrão dominante nos lares de hoje é semelhante ao do escritório médio há dez anos: muitos computadores que não se comunicam uns com os outros. No futuro, talvez seja comum a interligação do telefone, do televisor e de outros aparelhos eletrodomésticos em rede, permitindo que todos eles possam ser controlados remotamente. Sem dúvida, essa nova tecnologia dará origem a novas redes e novos protocolos.
Como exemplo da interconexão possível entre diferentes redes, considere a situação representada na Figura 5.42. Aqui temos uma rede corporativa com vários locais interligados por uma rede ATM geograficamente distribuída. Em um dos locais é usado um backbone óptico FDDI para conectar uma Ethernet, uma LAN sem fio 802.11 e a rede de mainframes SNA do centro de dados corporativo.

[arte: ver original p. 419]

[Dísticos]

[1] Mainframe

[2] Rede SNA

[3] Rede ATM

[4] Roteador

Conexão para a Internet

Ethernet

[5] Switch

[6] Anel FDDI

[7] Notebook

[8] 802.11

[F]Figura 5.42

[FL] Um conjunto de redes interconectadas

A finalidade de interconectar todas essas redes é permitir que usuários de qualquer delas se comuniquem com usuários de todas as outras, e também permitir que usuários de qualquer delas acessem dados armazenados em qualquer das redes. Alcançar esse objetivo significa enviar pacotes de uma rede para outra. Tendo em vista que freqüentemente as redes diferem em aspectos importantes, nem sempre é fácil transferir pacotes de uma rede para outra, como veremos agora.

[T3] 5.5.1 Diferenças entre redes

As redes podem diferir em várias aspectos. Algumas dessas diferenças, como técnicas de modulação ou formatos de quadros distintos, encontram-se nas camadas física e de enlace de dados. Essas diferenças não nos interessam agora. Em vez disso, na Figura 5.43 listamos algumas diferenças que podem ocorrer na camada de rede. E a superação dessas diferenças que torna a interligação de redes mais difícil do que a operação de uma única rede.

Quando os pacotes enviados por uma origem em uma rede devem transitar por uma ou mais redes externas antes de chegar à rede de destino (que também pode ser diferente da rede de origem), podem ocorrer muitos problemas nas interfaces

existentes entre as redes. Para começar, quando os pacotes de uma rede orientada a conexões têm de transitar por uma rede sem conexões, eles podem ser reordenados, fato que o transmissor não espera e com o qual o receptor não está preparado para lidar. Com freqüência, serão necessárias conversões de protocolos, o que talvez seja difícil caso a funcionalidade exigida não possa ser expressada. As conversões de endereços também serão necessárias, o que talvez exija algum tipo de sistema de diretórios. A passagem de pacotes de multidifusão por uma rede que não aceita esse recurso requer a geração de pacotes separados para cada destino.

Os diferentes tamanhos máximos de pacotes usados por redes distintas é uma grande dor de cabeça. Como passar um pacote de 8000 bytes por uma rede cujo tamanho máximo é de 1500 bytes? As diferentes qualidades de serviço se tornam um problema quando um pacote que tem restrições de entrega em tempo real passa por uma rede que não oferece qualquer garantia nesse sentido.

[arte: ver original p. 420]

[T]Tabela

Item	**Algumas possibilidades**
Serviço oferecido	Orientado a conexões e sem conexões
Protocolos	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Endereçamento	Simples (802) e hierárquico (IP)
Multidifusão	Presente ou ausente (também difusão)
Tamanho do pacote	Cada rede tem seu próprio tamanho máximo
Qualidade de serviço	Pode estar presente ou ausente; muitos tipos diferentes
Tratamento de erros	Confiável, entrega ordenada e entrega não ordenada
Controle de fluxo	Janela deslizante, controle de taxa, outros ou nenhum
Controle de congestionamento	Balde furado, balde de símbolos, RED, pacotes reguladores etc.

Segurança Regras de privacidade, criptografia etc.

Parâmetros Diferentes timeouts, especificações de fluxo etc.

Contabilidade Por tempo de conexão, por pacote, por byte ou nenhuma

[F]Figura 5.43

[FL] Algumas das muitas diferenças possíveis entre redes

Com freqüência, os controles de erro, fluxo e congestionamento apresentam diferenças entre as redes. Se a origem e o destino estiverem esperando que todos os pacotes sejam entregues em seqüência, sem erros, e uma rede intermediária simplesmente descartá-los ao pressentir uma possibilidade de congestionamento, muitas aplicações apresentarão falhas. O mesmo acontecerá se pacotes que estiverem vagando perdidos em um determinado momento forem eventualmente entregues a seu destino, se esse comportamento não tiver sido antecipado e tratado. Diferentes mecanismos de segurança, definições de parâmetro, regras de contabilidade e até mesmo leis de privacidade nacionais também podem causar problemas.

[T3] 5.5.2 Como as redes podem ser conectadas

As redes podem ser interconectadas por dispositivos diferentes, como vimos no Capítulo 4. Vamos rever rapidamente esse material. Na camada física, as redes podem ser conectadas por repetidores ou hubs, que apenas movem os bits de uma rede para uma rede idêntica. Em sua maioria, esses dispositivos são analógicos e não reconhecem nenhum aspecto dos protocolos digitais (eles simplesmente regeneram sinais).

Subindo uma camada, encontramos pontes e switches, que operem na camada de enlace de dados. Eles podem aceitar quadros, examinar os endereços MAC e encaminhar os quadros para uma rede diferente, enquanto executam uma

conversão de protocolos secundária no processo. Por exemplo, a conversão de Ethernet para FDDI ou 802.11.

Na camada de rede, temos roteadores que podem conectar duas redes. Se duas redes tiverem camadas de rede distintas, talvez o roteador seja capaz de realizar a conversão entre os formatos de pacotes, embora a conversão de pacotes agora seja cada vez mais rara. Um roteador que pode manipular vários protocolos é chamado **roteador multiprotocolo**.

Na camada de transporte, encontramos gateways de transporte, que podem fazer a interface entre duas conexões de transporte. Por exemplo, um gateway de transporte poderia permitir que os pacotes fluíssem entre uma rede TCP e uma rede SNA, que tem um protocolo de transporte diferente, essencialmente unindo uma conexão TCP a uma conexão SNA.

Por fim, na camada de aplicação, os gateways de aplicação convertem a semântica das mensagens. Como exemplo, gateways situados entre o correio eletrônico da Internet (RFC 822) e o correio eletrônico X.400 devem analisar as mensagens de correio eletrônico e alterar diversos campos de cabeçalho.

Neste capítulo, vamos nos concentrar na interligação de redes feita na camada de rede. Para ver a diferença entre essa interligação e a comutação na camada de enlace de dados, examine a Figura 5.44. Na Figura 5.44(a), a máquina de origem *S* deseja enviar um pacote à máquina de destino *D*. Essas máquinas estão em redes Ethernet diferentes, conectadas por um switch. *S* encapsula o pacote em um quadro e o transmite. O quadro chega ao switch, que determina então que o quadro tem seguir para a LAN 2, examinando seu endereço MAC. O switch simplesmente remove o quadro da LAN 1 e o deposita na LAN 2.

[arte: ver original p. 421]

[Dísticos]

[1] Legenda

Cabeçalho

Pacote

Final

[2] Switch

LAN 1 LAN 2

(a)

[3] Roteador

LAN 1 LAN 2

(b)

[F]Figura 5.44

[FL] (a) Duas redes Ethernet conectadas por um switch. (b) Duas redes Ethernet conectadas por roteadores

Agora, vamos considerar a mesma situação mas com as duas redes Ethernet conectadas por um par de roteadores, em vez de um switch. Os roteadores estão conectados por uma linha ponto a ponto, possivelmente uma linha dedicada com milhares de quilômetros. O quadro é extraído pelo roteador e o pacote é removido do campo de dados do quadro. O roteador examina o endereço no pacote (por exemplo, um endereço IP) e procura pelo endereço em sua tabela de roteamento. Com base nesse endereço, ele decide enviar o pacote ao roteador remoto, encapsulado potencialmente em um tipo diferente de quadro, dependendo do protocolo de linha. Na outra extremidade, o pacote é inserido no campo de dados de um quadro Ethernet e depositado na LAN 2.

Uma diferença essencial entre o caso comutado (ou com ponte) e o caso roteado é apresentada a seguir. Com um switch (ou uma ponte), o quadro inteiro é transportado, de acordo com seu endereço MAC. Com um roteador, o pacote é extraído do quadro e o endereço contido no pacote é usado com o objetivo de

definir para onde enviá-lo. Os switches não precisam reconhecer o protocolo da camada de rede que está sendo usado para comutar pacotes. Os roteadores precisam reconhecer esse protocolo.

[T3] 5.5.3 Circuitos virtuais concatenados

São possíveis dois estilos de interligação de redes: concatenação orientada a conexões de sub-redes de circuitos virtuais e um estilo de inter-rede de datagramas. Agora, vamos examinar esses dois estilos; porém, primeiro uma palavra de advertência. No passado, a maioria das redes (públicas) era orientada a conexões (e as redes frame relay, SNA, 802.16 e ATM ainda são desse tipo). Então, com a rápida aceitação da Internet, os datagramas passaram a ser elegantes. Entretanto, seria um erro imaginar que os datagramas são para sempre. Nesse ramo, só a mudança é eterna. Com a importância crescente das redes de multimídia, é provável que as redes orientadas a conexões voltem a ter maior importância em uma forma ou outra, pois é mais fácil garantir qualidade de serviço com conexões que sem conexões. Portanto, vamos dedicar algum espaço às redes orientadas a conexões.

No modelo de circuitos virtuais concatenados, mostrado na Figura 5.45, é estabelecida uma conexão para um host de uma rede distante, de forma semelhante ao modo como as conexões normalmente são estabelecidas. A sub-rede percebe que o destino é remoto e cria um circuito virtual até o roteador mais próximo à rede de destino. Em seguida, ela constrói um circuito virtual desse roteador até um **gateway exterior** (roteador multiprotocolo). Esse gateway registra a existência do circuito virtual em suas tabelas e continua a construir um outro circuito virtual até o roteador da próxima sub-rede. Esse processo continua até que o host de destino tenha sido alcançado.

[arte: ver original p. 422]

[Dísticos]

[1]Roteador multiprotocolo

[2]Roteador

[3]Host

[4]Circuitos virtuais concatenados fim a fim

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.45

[FL] Interligação de redes com o uso de circuitos virtuais concatenados

Quando os pacotes de dados começam a fluir ao longo do caminho, cada gateway retransmite os pacotes que chegam, fazendo a conversão entre formatos de pacotes e números de circuitos virtuais, se necessário. Obviamente, todos os pacotes de dados devem percorrer a mesma seqüência de gateway e, assim, chegar de forma ordenada. Conseqüentemente, os pacotes de um fluxo nunca são reordenados pela rede.

A principal característica dessa estratégia é que a seqüência de circuitos virtuais é estabelecida entre a origem e o destino passando por um ou mais gateways. Cada gateway mantém tabelas que informam quais circuitos virtuais passam por ele, para onde eles devem ser roteados e qual é o número do novo circuito virtual.

Esse esquema funciona melhor quando todas as redes têm aproximadamente as mesmas propriedades. Por exemplo, se todas as redes garantirem a entrega confiável dos pacotes da camada de rede, o fluxo entre a origem e o destino também será confiável, a menos que haja uma falha em algum trecho ao longo da rota. Da mesma forma, se nenhuma das redes garantir a entrega confiável, a concatenação dos circuitos virtuais também não será confiável. Por outro lado, se

a máquina de origem está em uma rede que garante entrega confiável, mas uma das redes intermediárias pode perder pacotes, isso significa que a concatenação alterou fundamentalmente a natureza do serviço.

Os circuitos virtuais concatenados também são comuns na camada de transporte. Particularmente, é possível criar um canal de bits usando, digamos, a SNA, que termine em um gateway e ter uma conexão TCP do gateway até o próximo gateway. Dessa forma, será possível criar um circuito virtual fim a fim, abrangendo diferentes redes e protocolos.

[T3] 5.5.4 Interligação de redes sem conexões

O modelo alternativo de inter-rede é o modelo de datagramas, mostrado na Figura 5.46. Nesse modelo, o único serviço que a camada de rede oferece à camada de transporte é a capacidade de inserir datagramas na sub-rede, e esperar que o melhor aconteça. A camada de rede não tem qualquer noção do que seja um circuito virtual, e muito menos do que seja uma concatenação entre eles. Esse modelo não exige que todos os pacotes pertencentes a uma conexão percorram a mesma seqüência de gateways. Na Figura 5.46, os datagramas enviados do host 1 ao host 2 são mostrados seguindo diferentes rotas na inter-rede. Em seguida, é tomada uma decisão de roteamento específica para cada pacote, talvez dependendo do tráfego no momento em que o pacote é enviado. Essa estratégia pode usar diversas rotas, podendo atingir uma largura de banda mais alta que o modelo de circuitos virtuais concatenados. Por outro lado, não há garantia de que os pacotes chegarão em ordem ao destino, supondo-se que não haverá problemas.

O modelo da Figura 5.46 não é tão simples quanto parece. Primeiro, se cada rede tem seu próprio protocolo da camada de rede, não é possível que um pacote de uma rede transite por outra. Alguém poderia imaginar que os roteadores

multiprotocolo realmente tentam fazer conversões de um formato para outro. No entanto, a não ser que os dois formatos sejam muito semelhantes e tenham os mesmos campos de informações, essas conversões sempre serão incompletas e, com freqüência, estarão condenados a fracassar. Por isso, raramente se utiliza a conversão.

Um outro problema, ainda mais sério, é o endereçamento. Imagine a seguinte situação: um host da Internet está tentando enviar um pacote IP para um host em uma rede SNA vizinha. Os endereços IP e SNA são diferentes. Seria necessário fazer um mapeamento entre endereços IP e SNA em ambos os sentidos. Além disso, o conceito do que é endereçável é diferente. No IP, os hosts (na realidade, placas de interface) têm endereços. Na SNA, entidades diferentes de hosts (por exemplo, dispositivos e hardware) também podem ter endereços. Na melhor das hipóteses, alguém teria de manter um banco de dados mapeando da melhor maneira possível todos esses endereços, mas esse mapeamento seria constantemente uma fonte de problemas.

[arte: ver original p. 424]

[Dísticos]

[1]Os pacotes viajam individualmente e podem seguir diferentes rotas

[2]Roteador

[3]Host

[4]Roteador multiprotocolo

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.46

[FL] Uma inter-rede sem conexões

Outra idéia seria projetar um pacote "inter-rede" universal e levar todos os roteadores a reconhecê-lo. Na verdade, essa estratégia é o IP — um pacote projetado para ser carregado por muitas redes. É claro que o IPv4 (o protocolo da Internet atual) expulsa todos os outros formatos do mercado, o IPv6 (o protocolo da Internet do futuro) não se impôs, e nada de novo foi inventado, embora a história tenha outras sugestões. Fazer todas as pessoas concordarem em usar um único formato é algo difícil, quando as empresas percebem a vantagem comercial de ter um formato patenteado controlado por elas.

Vamos recapitular rapidamente as duas estratégias de interligação de redes. Em essência, o modelo de circuitos virtuais concatenados proporciona as mesmas vantagens que o uso de circuitos virtuais em uma única sub-rede: é possível reservar buffers com antecedência, garantir uma ordem seqüencial, usar cabeçalhos mais curtos e evitar os problemas causados por duplicatas de pacotes atrasados.

O modelo também apresenta algumas desvantagens: é necessário um espaço de tabela nos roteadores para cada conexão aberta, não existe um roteamento alternativo para evitar áreas congestionadas e o roteador apresenta uma certa vulnerabilidade ao longo do caminho. Também é difícil, senão impossível, implementá-lo se uma das redes envolvidas for uma rede de datagramas não confiável.

As propriedades da estratégia de datagramas para interligação de redes são quase idênticas às de sub-redes de datagramas, ou seja, maior potencial para congestionamento, maior potencial de adaptação, maior robustez em relação a falhas de roteadores e necessidade de cabeçalhos mais longos. Uma inter-rede pode conter vários algoritmos de roteamento adaptativos, da mesma forma que uma única rede de datagramas.

Uma das principais vantagens da estratégia de datagramas para a interligação de

redes é que ela pode ser usada em sub-redes que não usam circuitos virtuais. Muitas LANs, redes móveis (por exemplo, as que funcionam em aeronaves e navios) e até mesmo WANs se enquadram nessa categoria. Quando uma inter-rede inclui um desses tipos de redes, pode haver sérios problemas, caso a estratégia de interligação de redes se baseie em circuitos virtuais.

[T3] 5.5.5 Tunneling

Lidar com a interligação de duas redes diferentes é extremamente difícil. Entretanto, existe um caso especial muito comum que proporciona bons resultados. Isso acontece quando os hosts de origem e de destino estão no mesmo tipo de rede, mas há uma rede de outro tipo entre eles. Por exemplo, imagine um banco internacional que tem uma Ethernet baseada no TCP/IP em Paris, outra rede Ethernet TCP/IP em Londres e uma WAN não IP (por exemplo, uma rede ATM) entre elas, como mostra a Figura 5.47.

[arte: ver original p. 425]

[Dísticos]

[1]Funciona como uma linha serial

[2]Roteador multiprotocolo

[3]Túnel

[4]WAN

[5]Ethernet em Londres

[6]Quadro Ethernet

[7]Cabeçalho

[8] Pacote IP dentro do campo de carga útil do pacote da WAN

[9] Quadro Ethernet

[10] Ethernet em Paris

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.47

[FL] Tunneling de um pacote de Paris a Londres

A solução para esse problema é uma técnica chamada **tunneling** (**tunelamento**). Para enviar um pacote IP ao host 2, o host 1 cria o pacote que contém o endereço IP do host 2, insere-o em um quadro Ethernet endereçado ao roteador multiprotocolo de Paris e o coloca na Ethernet. Quando obtém o quadro, o roteador multiprotocolo remove o pacote IP, insere-o no campo de carga útil do pacote da camada de rede da WAN e o envia ao endereço da WAN do roteador multiprotocolo de Londres. Em Londres, o roteador remove o pacote IP e o envia ao host 2 dentro de um quadro Ethernet.

A WAN pode ser considerada um grande túnel, que se estende de um roteador multiprotocolo a outro. O pacote IP simplesmente viaja de uma extremidade à outra do túnel protegido em sua bela caixa. Ele não precisa se preocupar com a WAN, nem os hosts das duas redes Ethernet. Somente o roteador multiprotocolo precisa reconhecer os pacotes IP e WAN. Na verdade, a distância entre a seção intermediária de um roteador multiprotocolo e a seção intermediária do outro funciona como uma linha serial.

Uma analogia pode tornar o processo de tunneling mais claro. Imagine uma pessoa dirigindo seu carro de Paris a Londres. Na França, o carro trafega em baixa velocidade, usando sua própria energia; no entanto, ao chegar ao Canal da Mancha, ele é colocado em um trem de alta velocidade e é transportado para a Inglaterra pelo Eurotúnel (não é permitido o tráfego de automóveis nesse túnel). Na realidade, o carro está sendo transportado como uma carga, conforme mostra a Figura 5.48. Na outra extremidade, o carro passa a transitar nas estradas

inglesas e continua a trafegar em velocidade baixa, com sua própria energia. Em uma rede externa, o tunneling de pacotes funciona da mesma forma.

[arte: ver original p. 426a]

[Dísticos]

[1] Carro

[2] Canal da Mancha

[3] Londres

[4] Via férrea

[5]Vagão de trem

[6]Paris

[F]Figura 5.48

[FL] Transportando um carro por um túnel entre a França e a Inglaterra

[T3] 5.5.6 Roteamento inter-redes

O roteamento através de uma inter-rede é semelhante ao roteamento em uma única sub-rede, mas há algumas outras complicações. Por exemplo, imagine a inter-rede da Figura 5.49(a), na qual cinco redes estão conectadas por seis roteadores (possivelmente multiprotocolo). É complicado elaborar um modelo de grafo dessa situação, pois cada roteador multiprotocolo pode ter acesso direto (ou seja, pode enviar pacotes) a todos os roteadores ligados a uma rede com a qual tenha conexão. Por exemplo, *B* na Figura 5.49(a) pode acessar diretamente *A* e *C* por meio da rede 2 e também *D* pela rede 3. Isso nos leva ao grafo da Figura 5.49(b).

[arte: ver original p. 426b]

[Dísticos]

[1]Rede

[2]Roteador (multiprotocolo)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.49

[FL] (a) Uma inter-rede. (b) Um grafo da inter-rede

Quando o grafo estiver pronto, poderão ser aplicados algoritmos de roteamento conhecidos, como os algoritmos com vetor de distância e por estado de enlace, ao conjunto de roteadores multiprotocolo. O resultado será um algoritmo de roteamento de dois níveis. Em cada rede é usado um **protocolo de gateway interior (interior gateway protocol)**; no entanto, entre as redes é usado um **protocolo de gateway exterior — exterior gateway protocol** ("gateway" é um termo antigo para "roteador"). Na verdade, como cada rede é independente, todas elas podem usar diferentes algoritmos. Por serem independentes umas das outras, com freqüência cada rede de uma inter-rede é denominada **sistema autônomo (AS — Autonomous System).**

Normalmente, um pacote típico de inter-rede sai de sua LAN em direção ao roteador multiprotocolo local (no cabeçalho da camada MAC). Ao chegar lá, o código da camada de rede decide para qual roteador multiprotocolo deve encaminhar o pacote, usando suas próprias tabelas de roteamento. Se o roteador puder ser alcançado através do protocolo de rede natural do pacote, este será diretamente transmitido para lá. Caso contrário, o pacote será enviado por tunneling, sendo encapsulado no protocolo exigido pela rede intermediária. Esse processo é repetido até que o pacote chegue à rede de destino.

Uma das diferenças entre o roteamento inter-redes e o roteamento intra-rede é que, em geral, o primeiro pode exigir que sejam cruzados fronteiras internacionais. Subitamente, várias leis entram em jogo como, por exemplo, as

rígidas leis de privacidade da Suécia sobre a exportação de dados pessoais de seus cidadãos. Outro exemplo é a lei canadense que afirma que o tráfego de dados originário do Canadá e que tem destino no Canadá não pode sair do país. Essa lei significa que o tráfego de Windsor, em Ontário, para Vancouver não pode ser roteado via Detroit, ainda que essa rota seja a mais rápida e a mais econômica.

Outra diferença entre roteamento interior e exterior é o custo. Em geral, quando a rede é simples, aplica-se um único algoritmo de tarifação. Entretanto, diferentes redes podem ser gerenciadas de formas distintas, e uma rota pode ser menos dispendiosa que outra. Da mesma forma, a qualidade do serviço oferecido por redes distintas pode ser diferente, e esse pode ser o motivo para se escolher uma rota em detrimento de outra.

[T3] 5.5.7 Fragmentação

Cada rede impõe um tamanho máximo a seus pacotes. Dentre as principais causas para essa limitação, temos:

1. Hardware (por exemplo, o tamanho de um quadro Ethernet).
2. Sistema operacional (por exemplo, todos os buffers têm 512 bytes).
3. Protocolos (por exemplo, o número de bits do campo de tamanho do pacote).
4. Compatibilidade com algum padrão (inter)nacional.
5. Desejo de reduzir de alguma forma as retransmissões provocadas por erros.
6. Desejo de evitar que um pacote ocupe o canal por muito tempo.

O resultado de todos esses fatores é que os projetistas de redes não têm liberdade para escolher o tamanho máximo de pacote que desejam. As cargas máximas variam de 48 bytes (células ATM) a 65.515 bytes (pacotes IP), apesar de o tamanho da carga útil em geral ser maior em camadas mais altas.

Um problema óbvio surge quando um pacote muito grande tem de trafegar por

uma rede cujo tamanho máximo de pacote é muito pequeno. Uma solução é garantir que o problema não ocorre de forma alguma. Em outras palavras, a inter-rede deve usar um algoritmo de roteamento que evite o envio de pacotes por redes que não sejam capazes de tratá-los. Entretanto, essa solução não é exatamente a melhor opção. O que acontecerá se o pacote original for muito grande para ser tratado pela rede de destino? Dificilmente o algoritmo de roteamento poderá ignorar o destino.

Basicamente, a única solução para o problema é permitir que os gateways dividam os pacotes em **fragmentos** enviando cada fragmento como um pacote de inter-rede separado. Entretanto, como todos os pais de filhos muito pequenos sabem, é bem mais fácil converter um objeto grande em pequenos fragmentos do que executar o processo inverso. (Os físicos deram um nome a esse efeito: a segunda lei da termodinâmica.) As redes de comutação de pacotes também têm problemas para juntar os fragmentos novamente.

[arte: ver original p. 428]

[Dísticos]

[1]Rede 1 Rede 2

Pacote

[2]G_1 G_2

G_1 divide um pacote grande em fragmentos G_2 volta a montar os fragmentos

(a)

[3]G_3 G_4

G_3 divide novamente um pacote grande em fragmentos G_4 volta a montar os fragmentos mais uma vez

[4] Pacote

G_1 G_2

G_1 divide um pacote grande em fragmentos

[5] G_3 G_4

Os fragmentos não são montados novamente até que o destino final (um host) seja alcançado

(b)

[F]Figura 5.50

[FL] (a) Fragmentação transparente. (b) Fragmentação não transparente

Existem duas estratégias opostas para recombinar os fragmentos e recompor o pacote original. A primeira estratégia é tornar a fragmentação causada por uma rede de "pequenos pacotes" transparente para todas as outras redes por onde o pacote deverá passar até chegar ao destino final. Essa opção é mostrada na Figura 5.50(a). Nessa abordagem, a rede de pequenos pacotes tem gateways (mas provavelmente, roteadores especializados) que fazem a interface com outras redes. Quando um pacote muito grande chega a um gateway, este o divide em fragmentos. Cada fragmento é endereçado ao mesmo gateway de saída no qual os fragmentos são recombinados. Dessa forma, a passagem de pequenos pacotes pela rede torna-se transparente. As redes subseqüentes nem tomam conhecimento de que ocorreu essa fragmentação. Por exemplo, as redes ATM dispõem de hardware especial para proporcionar uma fragmentação transparente dos pacotes em células e sua posterior recombinação em pacotes. No mundo das redes ATM, a fragmentação é chamada segmentação; o conceito é o mesmo, mas alguns detalhes são diferentes.

A fragmentação transparente é simples, mas apresenta alguns problemas. Por um lado, o gateway de saída deve saber quando recebeu todos os fragmentos; portanto, é necessário incluir um campo de contagem ou bit de "fim de pacote" em cada pacote. Por outro lado, todos os pacotes têm de sair pelo mesmo gateway. Como não é permitido que alguns fragmentos sigam uma rota até o

destino final e que outros fragmentos percorram uma rota distinta, há uma perda considerável em termos de desempenho. Um último problema é o overhead necessário para remontar e refragmentar repetidamente um pacote grande que passa por uma série de redes de pequenos pacotes. As redes ATM exigem fragmentação transparente.

A outra estratégia de fragmentação é evitar recombinar os fragmentos em gateways intermediários. Quando um pacote é fragmentado, cada fragmento é tratado como se fosse o pacote original. Todos os fragmentos passam pelo gateway (ou pelos gateways) de saída, como mostra a Figura 5.50(b). A recombinação só ocorre no host de destino. O IP funciona dessa maneira.

A fragmentação não transparente também apresenta alguns problemas. Por exemplo, ela exige que *todos* os hosts sejam capazes de fazer a remontagem. Outro problema é que, quando um pacote muito grande é fragmentado, o overhead total aumenta devido ao acréscimo de um cabeçalho a cada fragmento. Enquanto no primeiro método esse overhead desaparece assim que o fragmento sai da rede de pequenos pacotes, nesse método o overhead permanece igual até o fim do trajeto. Entretanto, uma vantagem da fragmentação não transparente é que agora é possível usar vários gateways de saída e obter um desempenho melhor. É óbvio que, se o modelo de circuitos virtuais concatenados estiver sendo usado, essa vantagem não terá qualquer utilidade.

Quando um pacote é fragmentado, os fragmentos devem ser numerados de forma que o fluxo de dados original possa ser reconstruído. Uma forma de numerar os fragmentos é usar uma árvore. Se o pacote 0 tiver de ser dividido, os fragmentos serão chamados de 0.0, 0.1, 0.2 etc. Se esses fragmentos tiverem de ser fragmentados posteriormente, os fragmentos resultantes serão numerados por 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 etc. Caso no cabeçalho tenham sido reservados campos suficientes para uma situação totalmente adversa e caso

nenhuma cópia tenha sido gerada, esse esquema será suficiente para garantir que todos os fragmentos serão corretamente remontados no destino, independente da ordem em que chegarem.

No entanto, mesmo que uma rede perca ou descarte pacotes, há necessidade de retransmissões fim a fim, cujos efeitos são desastrosos para o sistema de numeração. Suponha que um pacote de 1024 bits seja inicialmente dividido em quatro fragmentos de mesmo tamanho, 0.0, 0.1, 0.2 e 0.3. O fragmento 0.1 é perdido, mas as outras partes chegam ao destino. Eventualmente, a origem sofre um timeout e retransmite o pacote original. Dessa vez, a lei de Murphy ataca, e a rota seguida passa por uma rede com um limite de 512 bits; portanto, são gerados dois fragmentos. Quando o novo fragmento 0.1 chegar ao destino, o receptor concluirá que todos os quatro fragmentos foram recebidos e reconstruirá o pacote incorretamente.

Um sistema de numeração completamente diferente (e muito melhor) é fazer com que o protocolo da inter-rede defina um tamanho de fragmento elementar suficientemente pequeno para que esse fragmento possa passar por qualquer rede. Quando um pacote é fragmentado, todas as partes têm tamanho igual ao do fragmento elementar, com exceção do último, que pode ser mais curto. Um pacote de inter-rede pode conter vários fragmentos, por razões de eficiência. O cabeçalho da inter-rede deve fornecer o número de pacote original e o número do (primeiro) fragmento elementar contido no pacote. Em geral, também deve haver um bit indicando que o último fragmento elementar contido no pacote de inter-rede é o último do pacote original.

Essa estratégia requer dois campos de seqüência no cabeçalho da inter-rede: o número do pacote original e o número do fragmento. É claro que existe um compromisso entre o tamanho do fragmento elementar e o número de bits no número do fragmento. Como o tamanho do fragmento elementar é,

supostamente, aceitável para cada rede, a fragmentação subseqüente de um pacote de inter-rede contendo diversos fragmentos não causa problemas. O último limite aqui seria reduzir o fragmento elementar a um único bit ou byte, com o número do fragmento representando o deslocamento de byte ou bit no pacote original, como mostra a Figura 5.51.

[arte: ver original p. 430]

[Dísticos]

[1]Número do primeiro fragmento elementar nesse pacote

Número do pacote Fim do bit de pacote 1 byte

[2] Cabeçalho

(a)

[3] Cabeçalho Cabeçalho

(b)

[4]Cabeçalho Cabeçalho Cabeçalho

(c)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.51

[FL] A fragmentação quando o tamanho dos dados elementares é 1 byte. (a) O pacote original, contendo 10 bytes de dados. (b) Fragmentos depois da passagem por uma rede cujo tamanho máximo de pacote é 8 bytes de carga útil mais cabeçalho. (c) Fragmentos depois da passagem por um gateway de tamanho 5

Alguns protocolos de inter-redes levam esse método ainda mais longe e consideram toda a transmissão em um circuito virtual um único pacote gigantesco, de forma que cada fragmento contenha o número absoluto de byte

do primeiro byte contido no fragmento.

[T2] 5.6 A camada de rede na Internet

Antes de entrarmos nos detalhes específicos da camada de rede na Internet, vale a pena examinar os princípios que orientaram seu projeto no passado e que fazem da Internet o sucesso que é hoje. Com muita freqüência hoje em dia, as pessoas parecem ter esquecido esses princípios. Eles são enumerados e discutidos na RFC 1958, que vale a pena ler (e cuja leitura deve ser obrigatória para todos os projetistas de protocolos — com um exame de avaliação no final da leitura). Essa RFC se baseia intensamente em idéias encontradas em (Clark, 1988; e Saltzer *et al.*, 1984). Resumiremos agora aqueles que consideramos os 10 princípios fundamentais (do mais importante para o menos importante).

1. **Certifique-se de que funciona.** Não conclua o projeto ou o padrão até que vários protótipos tenham conseguido se comunicar com sucesso uns com os outros. Um número muito grande de projetistas escreve no início um padrão de 1000 páginas, obtém sua aprovação, e depois descobre que ele tem falhas profundas e que não funciona. Então, esse projetistas escrevem a versão 1.1 do padrão. Esse não é o melhor caminho.

2. **Mantenha a simplicidade.** Quando estiver em dúvida, use a solução mais simples. William de Occam enunciou esse princípio (a navalha de Occam) no Século 14. Em termos modernos: os recursos entram em conflito. Se um recurso não for absolutamente essencial, deixe-o de fora, em especial se o mesmo efeito puder ser obtido pela combinação de outros recursos.

3. **Faça escolhas claras.** Se houver várias maneiras de executar a mesma ação, escolha apenas uma. Ter duas ou mais opções para realizar a mesma ação é procurar problemas. Com freqüência, os padrões têm diferentes opções, modos ou parâmetros, porque várias partes poderosas insistem em afirmar que sua

alternativa é a melhor. Os projetistas devem resistir com firmeza a essa tendência. Basta dizer não.

4. **Explore a modularidade.** Esse princípio leva diretamente à idéia de pilhas de protocolos, em que cada uma das camadas é independente de todas as outras. Desse modo, se as circunstâncias exigirem mudanças em um módulo ou em uma camada, os outros itens não serão afetados.

5. **Espere heterogeneidade.** Diferentes tipos de hardware, instalações de transmissão e aplicações ocorrerão em qualquer rede de grande porte. Para lidar com isso, o projeto de rede deve ser simples, geral e flexível.

6. **Evite opções e parâmetros estáticos.** Se os parâmetros forem inevitáveis (por exemplo, tamanho máximo de pacote), é melhor fazer o transmissor e o receptor negociarem um valor do que definir opções fixas.

7. **Procure um bom projeto; ele não precisa ser perfeito.** Freqüentemente, os projetistas têm um bom projeto, mas não conseguem lidar com algum caso especial complicado. Em vez de alterar o projeto, os projetistas devem dar continuidade ao bom projeto e entregar o fardo de trabalhar com ele às pessoas que fizeram as exigências complexas.

8. **Seja rígido ao enviar e tolerante ao receber.** Em outras palavras, só envie pacotes que obedeçam rigorosamente aos padrões, mas espere receber pacotes que talvez não sejam plenamente compatíveis e procure lidar com eles.

9. **Pense na escalabilidade.** Se o sistema tiver de manipular milhões de hosts e bilhões de usuários de forma efetiva, nenhum banco de dados centralizado de qualquer tipo será tolerável, e a carga deverá ser dispersa da maneira mais uniforme possível pelos recursos disponíveis.

10. **Considere desempenho e custo.** Se uma rede tiver fraco desempenho ou custos exagerados, ninguém a usará.

Agora vamos deixar de lado os princípios gerais e iniciar o exame dos detalhes

da camada de rede da Internet. Na camada de rede, a Internet pode ser vista como um conjunto de sub-redes ou **sistemas autônomos** conectados entre si. Não existe uma estrutura real, mas diversos backbones principais, construídos a partir de linhas de grande largura de banda e roteadores rápidos. Conectadas aos backbones estão as redes regionais (nível médio), e conectadas a essas redes regionais estão as LANs de muitas universidades, empresas e provedores de serviços da Internet. Um esquema dessa organização semi-hierárquica é mostrado na Figura 5.52.

O elemento que mantém a Internet unida é o protocolo da camada de rede, o IP (Internet Protocol). Ao contrário da maioria dos protocolos da camada de rede mais antigos, o IP foi projetado desde o início tendo como objetivo a interligação de redes. Uma boa maneira de pensar na camada de rede é essa. A tarefa do IP é fornecer a melhor forma possível (ou seja, sem garantias) de transportar datagramas da origem para o destino, independente dessas máquinas estarem na mesma rede ou de haver outras redes entre elas.

Na Internet, a comunicação funciona da forma descrita a seguir. A camada de transporte recebe os fluxos de dados e os divide em datagramas. Teoricamente, cada datagrama pode ter até 64 Kbytes; no entanto, na prática, geralmente eles têm no máximo 1500 bytes (e portanto cabem em um único quadro Ethernet). Cada datagrama é transmitido pela Internet, talvez fragmentado em unidades menores durante o percurso até o destino. Quando todos os fragmentos finalmente chegam à máquina de destino, eles são remontados pela camada de rede no datagrama original. Em seguida, esse datagrama é entregue à camada de transporte, que o insere no fluxo de entrada do processo de recepção. Como podemos ver na Figura 5.52, um pacote originário do host 1 tem de passar por seis redes para chegar ao host 2. Na prática, esse número freqüentemente é muito maior que seis.

[arte: ver original p. 433]

[Dísticos]

[1] Linhas dedicadas para a Ásia

Um backbone norte-americano

[2] Linha transatlântica dedicada

Um backbone europeu

[3] Rede regional

A 1

LAN Ethernet IP

[4]C

Rede SNA

Túnel

D

LAN token ring IP

[5]Roteador IP

Rede nacional

B Host

2

LAN Ethernet IP

[F]Figura 5.52

[FL] A Internet é um conjunto de muitas redes interconectadas

[T3] 5.6.1 O protocolo IP

Um local apropriado para iniciar nosso estudo da camada de rede da Internet é o formato dos próprios datagramas IP. Um datagrama IP consiste em uma parte de cabeçalho e uma parte de texto. O cabeçalho tem uma parte fixa de 20 bytes e uma parte opcional de tamanho variável. O formato do cabeçalho é mostrado na

Figura 5.53. Ele é transmitido em uma ordem big endian: da esquerda para a direita, com o bit de mais alta ordem do campo *Version* aparecendo primeiro. (O SPARC é um big endian; o Pentium é um little endian.) Nas máquinas little endian, é necessária a conversão de software na transmissão e na recepção.

O campo *Version* controla a versão do protocolo à qual o datagrama pertence. Incluindo-se a versão em cada datagrama, é possível verificar a transição entre as versões, o que pode levar meses ou até mesmo anos, com algumas máquinas executando a versão antiga e outras executando a nova versão. Atualmente, está ocorrendo uma transição entre o IPv4 e o IPv6, que já acontece há anos e nem sequer está próxima de terminar (Durand, 2001; Wiljakka, 2002; e Waddington e Chang, 2002). Algumas pessoas pensam até que ela nunca acontecerá (Weiser, 2001). A propósito da numeração, o IPv5 foi um protocolo de fluxo em tempo real experimental, e nunca foi amplamente utilizado.

Como o tamanho do cabeçalho não é constante, existe um campo no cabeçalho, *IHL*, que informa seu tamanho em palavras de 32 bits. O valor mínimo é 5, quando não há nenhuma opção presente. O valor máximo desse campo de 4 bits é 15, o que limita o cabeçalho a 60 bytes e o campo *Options* a 40 bytes. Para algumas opções, como a que registra a rota percorrida pelo pacote, 40 bytes é muito pouco, o que torna a opção inútil.

[arte: ver original p. 434]

[Dísticos]

[1]32 Bits

[2]Version IHL Type of service Total length

[3]Identification DF MF Fragment offset

[4]Time to live Protocol Header checksum

[5]Source address

[6]Destination address

[7]Options (0 ou mais palavras)

[F]Figura 5.53

[FL] O cabeçalho IPv4 (Internet Protocol)

O campo *Type of service* é um dos poucos campos que tiveram seu significado (ligeiramente) modificado ao longo dos anos. Ele foi e ainda é destinado a distinguir entre diferentes classes de serviço. São possíveis várias combinações de confiabilidade e velocidade. Em se tratando de voz digitalizada, a entrega rápida vence a entrega segura. Para a transferência de arquivos, uma transmissão sem erros é mais importante do que uma transmissão rápida.

Originalmente, o campo de 6 bits continha (da esquerda para a direita) um campo *Precedence* de três bits e três flags, *D*, *T* e *R*. O campo *Precedence* tinha uma prioridade que variava de 0 (normal) a 7 (pacote de controle de rede). Os três bits de flags permitiam que o host especificasse o que era mais importante no conjunto {Retardo, Throughput, Confiabilidade}. Teoricamente, esses campos permitem que os roteadores optem, por exemplo, entre um enlace de satélite com alto throughput, mas com grande retardo ou uma linha dedicada com baixo throughput e baixo retardo. Na prática, os roteadores atuais ignoram completamente o campo *Type of service*.

Mais tarde, a IETF resolveu alterar ligeiramente o campo para acomodar serviços diferenciados. Seis dos bits são usados para indicar a que classe de serviço, dentre as classes discutidas antes, pertence cada pacote. Essas classes incluem as quatro prioridades de enfileiramento, três probabilidades de descarte e as classes históricas.

O campo *Total length* inclui tudo o que há no datagrama — cabeçalho e dados. O tamanho máximo é de 65.535 bytes. Atualmente, esse limite superior é tolerável, mas com as futuras redes de gigabits serão necessários datagramas maiores.

O campo *Identification* é necessário para permitir que o host de destino determine a qual datagrama pertence um fragmento recém-chegado. Todos os fragmentos de um datagrama contêm o mesmo valor de *Identification*.

Em seguida, há um bit não utilizado e dois campos de 1 bit. *DF* significa Don't Fragment (não fragmentar). Trata-se de uma ordem para os roteadores não fragmentarem o datagrama, porque a máquina de destino é incapaz de juntar os fragmentos novamente. Por exemplo, quando um computador é reinicializado, sua ROM solicita que uma imagem de memória seja enviada a ela como um único datagrama. Marcando-se o datagrama com o bit *DF*, o transmissor sabe que ele chegará em um único fragmento, mesmo que isso signifique que o datagrama deve evitar uma rede de pacotes pequenos que esteja no melhor caminho, e seguir por uma rota menos adequada. Todas as máquinas são obrigadas a aceitar fragmentos de 576 bytes ou menos.

MF significa More Fragments (mais fragmentos). Todos os fragmentos, exceto o último, têm esse conjunto de bits, necessário para se saber quando chegaram todos os fragmentos de um datagrama.

O campo *Fragment offset* informa a que ponto do datagrama atual o fragmento pertence. Todos os fragmentos de um datagrama, com exceção do último, devem ser múltiplos de 8 bytes, a unidade elementar de fragmento. Como são fornecidos 13 bits, existem no máximo 8192 fragmentos por datagrama, resultando em um tamanho máximo de datagrama igual a 65.536 bytes, um a mais que o campo *Total length*.

O campo *Time to live* é um contador usado para limitar a vida útil dos pacotes. Esse campo conta o tempo em segundos, permitindo uma vida útil máxima de 255 s. Esse contador deve ser decrementado a cada hop e supõem-se que ele seja decrementado diversas vezes quando estiver enfileirado durante um longo tempo em um roteador. Na prática, ele simplesmente conta os hops. Quando o

contador chega a zero, o pacote é descartado e um pacote de advertência é enviado ao host de origem. Esse recurso evita que os datagramas fiquem vagando indefinidamente, algo que aconteceria se as tabelas de roteamento fossem danificadas.

Quando tiver montado um datagrama completo, a camada de rede precisará saber o que fazer com ele. O campo *Protocol* informa a que processo de transporte o datagrama deve ser entregue. O TCP é uma possibilidade, mas também há o UDP e alguns outros. A numeração dos protocolos se aplica a toda a Internet. Os protocolos e outros números atribuídos foram listados inicialmente na RFC 1700, mas hoje eles estão contidos em um banco de dados on-line localizado em *www.iana.org*.

O campo *Header checksum* confere apenas o cabeçalho. Esse total de verificação é útil para a detecção de erros gerados por palavras de memória incorretas em um roteador. O algoritmo tem como objetivo somar todas as meias palavras de 16 bits à medida que elas chegam, utilizando a aritmética de complemento de 1 e depois calculando o complemento de 1 do resultado. Para os propósitos desse algoritmo, supomos que o campo *Header checksum* seja zero no momento da chegada. O uso do algoritmo é mais eficaz do que uma soma normal. Observe que *Header checksum* deve ser recontado a cada hop, porque pelo um campo sempre se altera (o campo *Time to live*), mas existem artifícios que podem ser usados para acelerar o cálculo.

Os campos *Source address* e *Destination address* indicam o número da rede e o número do host. Discutiremos os endereços da Internet na próxima seção. O campo *Options* foi projetado para permitir que versões posteriores do protocolo incluam informações inexistentes no projeto original, possibilitando a experimentação de novas idéias e evitando a alocação de bits de cabeçalho para informações raramente necessárias. Existem opções de tamanhos variáveis. Cada

uma começa com um código de um byte identificando a opção. Algumas opções são seguidas por um campo de tamanho de opção de 1 byte e depois por um ou mais bytes de dados. O campo *Options* é preenchido até alcançar um múltiplo de quatro bytes. Originalmente, havia cinco opções definidas, como mostra a Figura 5.54, mas desde então foram acrescentadas mais algumas. Agora, a lista completa é mantida on-line em *www.iana.org/assignments/ip-parameters*.

[arte: ver original p. 436]

[T]Tabela

Opção	**Descrição**
Security	Especifica o nível de segurança do datagrama
Strict source routing	Mostra o caminho completo a ser seguido
Loose source routing	Apresenta uma lista de roteadores que não devem ser esquecidos
Record route	Faz com que cada roteador anexe seu endereço IP
Timestamp	Faz com que cada roteador anexe seu endereço e seu timbre de hora

[F]Figura 5.54

[FL] Algumas opções do IP

A opção *Security* mostra o nível de segurança da informação. Teoricamente, um roteador militar poderia usar esse campo para especificar que não se deve seguir rotas que passam por certos países que os militares consideram "mal comportados". Na prática, todos os roteadores a ignoram, pois a sua única função prática é ajudar os espiões a descobrir mais facilmente onde estão as melhores informações.

A opção *Strict source routing* fornece o caminho completo da origem ao destino como uma seqüência de endereços IP. O datagrama é obrigado a seguir exatamente essa rota. Essa opção é mais útil principalmente para os gerentes de

sistemas enviarem pacotes de emergência quando as tabelas de roteamento estão danificadas ou para fazer medições de sincronização.

A opção *Loose source routing* exige que o pacote percorra uma lista de roteadores específicos, na ordem determinada, mas permite que ele passe por outros roteadores durante o percurso. Normalmente, essa opção forneceria um pequeno número de roteadores, a fim de forçar um determinado caminho. Por exemplo, se for necessário forçar um pacote a ir de Londres até Sydney pelo oeste e não pelo leste, essa opção poderia especificar roteadores em Nova York, Los Angeles e Honolulu. Essa opção é útil principalmente quando considerações políticas ou econômicas exigem a passagem por certos países ou que eles sejam evitados.

A opção *Record route* informa aos roteadores ao longo do caminho que eles devem anexar seu endereço IP ao campo de opções. Isso permite que administradores de sistemas depurem algoritmos de roteamento. ("Por que os pacotes de Houston para Dallas estão passando primeiro por Tóquio?") Quando a ARPANET foi criada, nenhum pacote passava por mais de nove roteadores; por isso, 40 bytes de opção eram suficientes. Como mencionamos antes, agora esse espaço é muito pequeno.

Por fim, a opção *Timestamp* é semelhante à opção *Record route*, exceto pelo fato de além de registrar seu endereço IP de 32 bits, cada roteador também registrar um timbre de hora de 32 bits. Essa opção também se destina, principalmente, à depuração de algoritmos de roteamento.

[T3] 5.6.2 Endereços IP

Na Internet, cada host e cada roteador tem um endereço IP que codifica seu número de rede e seu número de host. A combinação é exclusiva: em princípio, duas máquinas na Internet nunca têm o mesmo endereço IP. Todos os endereços

IP têm 32 bits e são usados nos campos *Source address* e *Destination address* dos pacotes IP. É importante observar que um endereço IP não se refere realmente a um host. Na verdade, ele se refere a uma interface de rede; assim, se um host estiver em duas redes, ele precisará ter dois endereços IP. Porém, na prática, a maioria dos hosts está em uma única rede e, portanto, só tem um endereço IP.

Por várias décadas, os endereços IP foram divididos nas cinco categorias listadas na Figura 5.55. Essa alocação chegou a ser chamada **endereçamento de classe completo.** Embora não seja mais usada, ainda são comuns referências a essa alocação na literatura. Descreveremos em breve a substituição do endereçamento de classe completo.

[arte: ver original p. 437]

[Dísticos]

[1]32 Bits

Classe Intervalo de endereços de hosts

[2]A 0 Rede Host 1.0.0.0 a 127.255.255.255

[3]B 10 Rede Host 128.0.0.0 a 191.255.255.255

[4]C 110 Rede Host 192.0.0.0 a 223.255.255.255

[5]D 1110 Endereço de multidifusão 224.0.0.0 a 239.255.255.255

[6]E 1111 Reservado para uso futuro 240.0.0.0 a 247.255.255.255

[F]Figura 5.55

[FL] Formatos de endereços IP

Os formatos das classes A, B, C e D permitem até 128 redes com 16 milhões de hosts cada, 16.384 redes com hosts de até 64 K, 2 milhões de redes (por exemplo, LANs) com até 256 hosts cada (embora algumas dessas redes sejam especiais). Além disso, é admitida a multidifusão, na qual um datagrama é

direcionado a vários hosts. Os endereços que começam com 1111 são reservados para uso futuro. Atualmente, há 500.000 redes conectadas à Internet, e esse número cresce a cada ano. Os números de redes são atribuídos por uma corporação sem fins lucrativos chamada **ICANN (Internet Corporation for Assigned Names and Numbers)** para evitar conflitos. Por sua vez, a ICANN tem partes delegadas do espaço de endereços para diversas autoridades regionais, e estas fazem a doação de endereços IP a ISPs e outras empresas.

Em geral, os endereços de rede, que são números de 32 bits, são escritos em **notação decimal com pontos.** Nesse formato, cada um dos 4 bytes é escrito em notação decimal, de 0 a 255. Por exemplo, o endereço hexadecimal de 32 bits C0290614 é escrito como 192.41.6.20. O endereço IP mais baixo é 0.0.0.0 e o mais alto é 255.255.255.255.

Os valores 0 e -1 (todos os dígitos 1) têm significados especiais, como mostra a Figura 5.56. O valor 0 significa esta rede ou este host. O valor -1 é usado como um endereço de difusão que significa todos os hosts na rede indicada.

[arte: ver original p. 438]

[Dísticos]

[1]Este host

[2] Host Um host nesta rede

[3] Difusão na rede local

[4] Rede

[5] Difusão em uma rede distante

[6] (Qualquer coisa)

[7] Loopback

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.56

[FL] Endereços IP especiais

O endereço IP 0.0.0.0 é usado pelos hosts quando eles estão sendo inicializados. Os endereços IP que têm 0 como número de rede se referem à rede atual. Esses endereços permitem que as máquinas façam referência às suas próprias redes sem saber seu número (mas elas precisam conhecer sua classe para saber quantos zeros devem ser incluídos). O endereço que consiste apenas em dígitos 1 permite a difusão na rede local, que em geral é uma LAN. Os endereços com um número de rede apropriado e que tiverem apenas valores 1 no campo de host permitem que as máquinas enviem pacotes de difusão para LANs distantes, em qualquer parte da Internet (embora muitos administradores de redes desativem esse recurso). Por fim, todos os endereços com o formato 127.*xx.yy.zz* são reservados para teste de loopback. Os pacotes enviados para esse endereço não são transmitidos; eles são processados localmente e tratados como pacotes de entrada. Isso permite que os pacotes sejam enviados para a rede local, sem que o transmissor saiba seu número.

[T4] Sub-redes

Como vimos, todos os hosts de uma rede devem ter o mesmo número de rede. Essa propriedade do endereçamento IP poderá causar problemas à medida que as redes crescem. Por exemplo, imagine uma universidade que começou com uma rede da classe B usada pelo departamento de ciência da computação para os computadores em sua Ethernet. Um ano mais tarde, o departamento de engenharia elétrica quis entrar na Internet, e assim comprou um repetidor para estender a rede Ethernet do departamento de ciência da computação até seu edifício. Como o tempo, muitos outros departamentos adquiriram computadores,

e o limite de quatro repetidores por rede Ethernet logo foi alcançado. Tornou-se necessária uma organização diferente.

Seria difícil obter um segundo endereço de rede, pois os endereços de rede são escassos, e a universidade já tinha endereços suficientes para mais de 60.000 hosts. O problema é a regra segundo a qual um único endereço da classe A, B ou C se refere a uma rede, e não a um conjunto de LANs. À medida que mais e mais organizações se encontravam nessa situação, era feita uma pequena mudança no sistema de endereçamento para lidar com ela.

A solução para esses problemas é permitir que uma rede seja dividida em diversas partes para uso interno, mas externamente continue a funcionar como uma única rede. Hoje, uma rede de campus típica seria semelhante à da Figura 5.57, com um roteador principal conectado a um ISP ou a uma rede regional e numerosas redes Ethernet espalhadas pelo campus em diferentes departamentos. Cada uma das redes Ethernet tem seu próprio roteador conectado ao roteador principal (possivelmente por meio de uma LAN de backbone, mas a natureza da conexão entre roteadores não é relevante nesse caso).

[arte: ver original p. 439]

[Dísticos]

[1] Roteador

Arte

Inglês

Francês

Música

[2] Para ISP

Roteador principal

[3] PC

Ciência da computação

Engenharia elétrica

Matemática

Física

Ethernet

[F]Figura 5.57

[FL] Uma rede de campus consistindo em LANs para vários departamentos

Na literatura sobre Internet, as partes da rede (nesse caso, redes Ethernet) são chamadas **sub-redes**. Como mencionamos no Capítulo 1, essa acepção apresenta conflitos com o termo "sub-rede" que significa o conjunto formado por todos os roteadores e linhas de comunicação de uma rede. Esperamos que o contexto esclareça o significado pretendido. Nesta seção e na próxima, a nova definição será a única a ser usada.

Quando um pacote entra no roteador principal, como este sabe para qual sub-rede (Ethernet) deve entregar o pacote? Uma alternativa seria uma tabela com 65.536 entradas no roteador principal, informando que roteador usar para cada host no campus. Essa idéia funcionaria, mas iria exigir uma tabela muito grande no roteador principal e um grande volume de manutenção manual, à medida que os hosts fossem acrescentados, movidos ou retirados de serviço.

Em vez disso, foi criado um esquema diferente. Basicamente, em vez de ter um único endereço da classe B com 14 bits para indicar o número da rede e 16 bits para indicar o número do host, alguns bits são retirados do número do host para criar um número de sub-rede. Por exemplo, se a universidade tivesse 35 departamentos, ela poderia usar um número de sub-rede de 6 bits e um número de host de 10 bits, permitindo até 64 redes Ethernet, cada uma com o máximo de 1022 hosts (0 e -1 não estão disponíveis, conforme mencionamos antes). Essa divisão poderia ser alterada mais tarde, caso ela se mostrasse incorreta.

Para implementar a divisão em sub-redes, o roteador principal precisa de uma **máscara de sub-rede** que indique a divisão entre o número de rede + sub-rede e o host, como mostra a Figura 5.58. As máscaras de sub-redes também são escritas em notação decimal com pontos, com a inclusão de uma barra vertical seguida pelo número de bits na parte de rede + sub-rede. No exemplo da Figura 5.58, a máscara de sub-rede pode ser escrita como 255.255.252.0. Uma notação alternativa é /22 para indicar que a máscara de sub-rede tem 22 bits.

[arte: ver original p. 440]

[Dísticos]

[1]32 bits

[2] Máscara de sub-rede

[3] Rede

[4] Sub-rede

[5] Host

[F]Figura 5.58

[FL] Uma rede da classe B dividida em 64 sub-redes

Fora da rede, a divisão em sub-redes não é visível; assim, a alocação de uma nova sub-rede não exige a intervenção da ICANN ou a mudança de quaisquer bancos de dados externos. Nesse exemplo, a primeira sub-rede pode usar os endereços IP a partir de 130.50.4.1, a segunda sub-rede pode se iniciar em 130.50.8.1, a terceira sub-rede pode começar em 130.50.12.1 e assim por diante. Para ver por que as sub-redes estão sendo contadas de quatro em quatro, observe que os endereços binários correspondentes são:

Sub-rede 1: 10000010 00110010 000001|00 00000001

Sub-rede 2: 10000010 00110010 000010|00 00000001

Sub-rede 3: 10000010 00110010 000011|00 00000001

Aqui a barra vertical (|) mostra o limite entre o número da sub-rede e o número de host. À sua esquerda está o número de sub-rede de 6 bits; à sua direita está o número de host de 10 bits.

Para ver como as sub-redes funcionam, é necessário explicar como os pacotes IP são processados em um roteador. Cada roteador tem uma tabela que lista algum número de endereços IP (rede, 0) e uma série de endereços IP (para essa rede ou host). O primeiro tipo informa como chegar a redes distantes. O segundo, como chegar a hosts locais. Associadas a essa tabela estão a interface de rede usada para alcançar o destino e algumas outras informações.

Quando um pacote IP é recebido, seu endereço de destino é procurado na tabela de roteamento. Se o destino for uma rede distante, o pacote será encaminhado para o próximo roteador da interface fornecida na tabela. Caso o destino seja um host local (por exemplo, na LAN do roteador), o pacote será enviado diretamente para lá. Se a rede não estiver presente, o pacote será enviado para um roteador predefinido que tenha tabelas maiores. Esse algoritmo significa que cada roteador só precisa controlar as outras redes e hosts locais, deixando de lado os pares (rede, host), o que reduz muito o tamanho da tabela de roteamento.

Quando a divisão em sub-redes é introduzida, as tabelas de roteamento são alteradas acrescentando-se entradas da forma (esta rede, sub-rede, 0) e (esta rede, esta sub-rede, host). Sendo assim, um roteador da sub-rede *k* sabe como alcançar todas as outras sub-redes, e também como chegar a todos os hosts da sub-rede *k*. Ele não precisa saber detalhes sobre os hosts de outras sub-redes. Na realidade, a única modificação é fazer com que cada roteador seja submetido a um AND booleano com a máscara de sub-rede, a fim de eliminar o número do host e pesquisa o endereço resultante em suas tabelas (depois de determinar qual é a classe da rede). Por exemplo, um pacote endereçado a 130.50.15.6 recebido no roteador principal passa pela operação AND booleana com a máscara

de sub-rede 255.255.252.0/22 para gerar o endereço 130.50.12.0. Esse endereço é usado para acessar as tabelas de roteamento com a finalidade de descobrir que linha de entrada usar para chegar ao roteador correspondente à sub-rede 3. Desse modo, a divisão em sub-redes reduz o espaço na tabela do roteador, criando uma hierarquia de três níveis que consiste em rede, sub-rede e host.

[T4] CIDR — Classless InterDomain Routing

O IP vem sendo amplamente utilizado há décadas. Ele tem funcionado muito bem, o que é demonstrado pelo crescimento exponencial da Internet. Infelizmente, o IP está se tornando uma vítima de sua própria popularidade, pois está ficando sem endereços. Esse enorme desastre causou muita discussão e controvérsia na comunidade da Internet sobre o que fazer em relação a ele. Nesta seção, descreveremos o problema e diversas soluções propostas para solucioná-lo.

Em 1987, alguns visionários previram que algum dia a Internet chegaria a 100.000 redes. Muitos especialistas desdenharam, dizendo que isso só aconteceria após muitas décadas, se acontecesse. A centésima milésima rede foi conectada em 1996. O problema, como mencionamos antes, é que a Internet está esgotando com rapidez os endereços IP disponíveis. Em princípio, existem mais de 2 bilhões de endereços, mas a prática de organizar o espaço de endereços por classes (ver Figura 5.55) faz com que milhões deles sejam desperdiçados. Particularmente, o verdadeiro vilão é a rede da classe B. Para muitas empresas, uma rede da classe A, com 16 milhões de endereços, é muito grande, e uma rede da classe C, com 256 endereços, é muito pequena. Uma rede da classe B, com 65.536 endereços, é a melhor solução. No folclore da Internet, essa situação é conhecida como **problema dos três ursos** (como na história infantil *Cachinhos de Ouro e os Três Ursos*).

Na realidade, um endereço da classe B é grande demais para a maioria das organizações. Estudos demonstraram que mais da metade de todas as redes da classe B têm menos de 50 hosts. Uma rede da classe C funcionaria muito bem nesse caso, mas não há dúvida de que todas as empresas que solicitaram um endereço da classe B pensaram que um dia ultrapassariam o campo dos hosts de 8 bits. Teria sido muito melhor fazer com que as redes da classe C utilizassem 10 bits, em vez de oito para o número de host, permitindo 1022 hosts por rede. Se isso tivesse acontecido, a maioria das empresas provavelmente teria optado por uma rede da classe C, e haveria meio milhão dessas redes (contra somente 16.384 redes da classe B).

É difícil culpar os projetistas da Internet por não terem fornecido mais (e menores) endereços da classe B. Na época em que foi tomada a decisão de criar as três classes, a Internet era uma rede de pesquisa conectando as principais universidades de pesquisa dos Estados Unidos (além de um número muito pequeno de empresas e instalações militares que realizavam pesquisas na área de redes). Até então, ninguém via a Internet como um sistema de comunicação do mercado de massa, rivalizando com a rede telefônica. Na época, alguém provavelmente diria: "Os Estados Unidos têm mais de 2000 faculdades e universidades. Mesmo que todas elas se conectassem à Internet, bem como muitas universidades de outros países, nunca chegaríamos a 16.000, pois não existem tantas universidades no mundo inteiro. Além disso, fazer do número de host um número inteiro de bytes acelera o processamento de pacotes."

Entretanto, se a divisão tivesse alocado 20 bits para o número de rede da classe B, outro problema teria surgido: a explosão da tabela de roteamento. Do ponto de vista dos roteadores, o espaço de endereços IP tem uma hierarquia de dois níveis, com números de rede e números de host. Os roteadores não precisam conhecer todos os hosts, mas têm de conhecer todas as redes. Se meio milhão de redes da

classe C estivessem em uso, cada roteador da Internet precisaria de uma tabela com meio milhão de entradas, uma por rede, indicando qual linha deveria ser usada para se chegar a uma determinada rede, além de outras informações.
O armazenamento físico de tabelas com meio milhão de entradas provavelmente é viável, embora dispendiosa para roteadores críticos que mantêm a tabela em RAM estática localizada em placas de E/S. O problema mais sério é que a complexidade de diversos algoritmos relacionados ao gerenciamento das tabelas cresce com uma rapidez maior que a linear. Pior ainda, grande parte do software e do firmware de roteadores existentes foi projetada em uma época em que a Internet tinha 1000 redes conectadas, e 10.000 redes pareciam algo muito distante. Muitas vezes, opções de projeto escolhidas nessa época se mostraram bem abaixo do nível considerado ótimo.
Além disso, diversos algoritmos de roteamento exigem que cada roteador transmita suas tabelas periodicamente (por exemplo, protocolos com vetor de distância). Quanto maiores as tabelas, maior será a probabilidade de que algumas partes se percam pelo caminho, resultando em dados incompletos na outra extremidade e talvez em instabilidades de roteamento.
O problema da tabela de roteamento poderia ter sido solucionado pelo estabelecimento de uma hierarquia mais profunda. Por exemplo, fazer com que cada endereço IP contenha um campo de país, estado/província, cidade, rede e host poderia funcionar. Nesse caso, cada roteador só precisaria saber como chegar a cada país, aos estados ou províncias de seu próprio país, às cidades de seu estado ou província e às redes de sua cidade. Infelizmente, essa solução exigiria muito mais de 32 bits para endereços IP e não utilizaria os endereços de uma forma eficiente (Liechtenstein teria tantos bits quanto os Estados Unidos).
Em resumo, algumas soluções resolvem um problema, mas criam outro. A solução implementada e que deu à Internet um pouco de espaço extra para

respirar foi o **CIDR (Classless InterDomain Routing).** A idéia básica por trás do CIDR, descrito na RFC 1519, é alocar os endereços IP restantes em blocos de tamanho variável, sem levar em consideração as classes. Se um site precisar, digamos, de 2000 endereços, ele receberá um bloco de 2048 endereços em um limite de 2048 bytes.

A eliminação das classes torna o encaminhamento mais complicado. No antigo sistema de classes, o encaminhamento funcionava da maneira descrita a seguir. Quando um pacote chegava a um roteador, uma cópia do endereço IP era deslocada 28 bits para a direita, a fim de gerar um número de classe de 4 bits. Em seguida, um desvio de 16 vias ordenava os pacotes em A, B, C e D (se houvesse suporte), com oito dos casos correspondendo à classe A, quatro à classe B, dois à classe C e ainda um caso para D e um para E. O código para cada classe era então usado para mascarar o número de rede de 8, 16 ou 24 bits e o alinhava à direita em uma palavra de 32 bits. Em seguida, o número de rede era pesquisado na tabela de A, B ou C, em geral pela indexação para as redes A e B e por hash para as redes C. Depois que a entrada era encontrada, a linha de saída podia ser pesquisada e o pacote era encaminhado.

Com o CIDR, esse algoritmo simples não funciona mais. Em vez disso, cada entrada de tabela de roteamento é estendida com uma máscara de 32 bits. Desse modo, agora existe uma única tabela de roteamento para todas as redes, consistindo em um array de triplas (endereço IP, máscara de sub-rede, linha de saída). Quando um pacote chega, seu endereço IP de destino é extraído. Depois (conceitualmente), a tabela de roteamento é varrida entrada por entrada, mascarando-se o endereço de destino e comparando-se esse endereço com a entrada de tabela, em busca de uma correspondência. É possível que várias entradas (com diferentes comprimentos de máscaras de sub-redes) correspondam e, nesse caso, será usada a máscara mais longa. Portanto, se

houver uma correspondência para a máscara /20 e uma máscara /24, será usada a entrada /24.

Foram criados algoritmos complexos para acelerar o processo de comparação de endereços (Ruiz-Sanchez *et al.*, 2001). Os roteadores comerciais utilizam chips VLSI personalizados com esses algoritmos incorporados em hardware.

Para tornar o algoritmo de encaminhamento mais fácil de entender, utilizaremos um exemplo no qual estão disponíveis milhões de endereços, começando em 194.24.0.0. Suponha que a Universidade de Cambridge precise de 2048 endereços e receba os endereços de 194.24.0.0 a 194.24.7.255 junto com a máscara 255.255.248.0. Em seguida, a Universidade de Oxford solicita 4096 endereços. Como um bloco de 4096 endereços deve ficar em um limite de 4096 bytes, não podem ser fornecidos endereços que comecem em 194.24.8.0. Em vez disso, são fornecidos endereços de 194.24.16.0 a 194.24.31.255, juntamente com a máscara 255.255.240.0. Agora, a Universidade de Edinburgh solicita 1024 endereços, e são atribuídos a ela os endereços de 194.24.8.0 a 194.24.11.255, bem como a máscara 255.255.252.0. Essas atribuições estão resumidas na Figura 5.59.

[arte: ver original p. 443]

[T]Tabela

Universidade	**Primeiro endereço**	**Último endereço**	**Quantidade**	**Escritos como**
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Disponível)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

[F]Figura 5.59

[FL] Um conjunto de atribuições de endereços IP

As tabelas de roteamento do mundo inteiro agora estão atualizadas com as três entradas atribuídas. Cada entrada contém um endereço básico e uma máscara de sub-rede. Essas entradas (em código binário) são as seguintes:

Endereço Máscara

C: 11000010 00011000 00000000 00000000 11111111 11111111 11111000 00000000

E: 11000010 00011000 00010000 00000000 11111111 11111111 11110000 00000000

O: 11000010 00011000 00010000 00000000 11111111 11111111 11110000 00000000

Agora, considere o que acontece quando um pacote chega endereçado a 194.24.17.4 que, em código binário, é representado por este string de 32 bits:

11000010 00011000 00010001 00000100

Primeiro, ele passa por uma operação AND booleana com a máscara de Cambridge, obtendo-se:

11000010 00011000 00010000 00000000

Esse valor não corresponde ao endereço básico de Cambridge; portanto, o endereço original passa por mais uma operação AND com a máscara de Edinburgh, transformando-se em:

11000010 00011000 00010000 00000000

Esse valor não corresponde ao endereço básico de Edinburgh; assim, experimenta-se a mesma operação AND com a máscara de Oxford, gerando:

11000010 00011000 00010000 00000000

Esse valor corresponde ao endereço básico de Oxford. Se não for encontrada nenhuma outra correspondência mais abaixo na tabela, será utilizada a entrada de Oxford, e o pacote será enviado pela linha especificada.

Agora vamos observar essas três universidades do ponto de vista de um roteador em Omaha, Nebraska, que tem apenas quatro linhas de saída: Minneapolis, Nova York, Dallas e Denver. Ao receber as três novas entradas, o software do roteador verifica que pode combinar todas as três entradas em uma única **entrada agregada** 194.24.0.0/19 com um endereço binário e a submáscara a seguir:

11000010 0000000 00000000 00000000 11111111 11111111 11100000 00000000

Essa entrada envia todos os pacotes destinados a quaisquer das três universidades para Nova York. Agregando as três entradas, o roteador de Omaha reduziu seu tamanho de tabela em duas entradas.

Se Nova York tem uma única linha para Londres, relativa a todo tráfego do Reino Unido, também se pode usar uma entrada agregada. Porém, se Nova York tiver linhas separadas para Londres e Edinburgh, serão necessárias três entradas separadas. A agregação é muito utilizada em toda a Internet para reduzir o tamanho das tabelas de roteadores.

Como uma nota final sobre esse exemplo, a entrada de rotas agregadas em Omaha também envia pacotes correspondentes aos endereços não atribuídos para Nova York. Desde que os endereços sejam de fato não atribuídos, isso não importa, porque eles não devem ocorrer. No entanto, se eles forem atribuídos mais tarde a uma empresa na Califórnia, será preciso uma entrada adicional, 194.24.12.0/22, para lidar com eles.

[T4] NAT — Network Address Translatiom

Os endereços IP são escassos. Um ISP poderia ter um endereço /16 (anteriormente da classe B), fornecendo 65.534 números de hosts. Se ele tiver um número maior do que esse de clientes, haverá um problema. Para os clientes individuais com conexões de discagem, uma forma de contornar o problema é

atribuir dinamicamente um endereço IP ao computador quando ele se conectar e efetuar logon, tomando o endereço IP de volta quando a sessão terminar. Desse modo, um único endereço /16 poderá manipular até 65.534 usuários ativos, o que provavelmente deve ser bastante bom para um ISP com várias centenas de milhares de clientes. Quando a sessão for encerrada, o endereço IP será designado novamente para outro usuário. Embora essa estratégia funcione bem no caso de um ISP com um número moderado de usuários domésticos, ela falha para ISPs que atendem principalmente a clientes de negócios.

O problema é que os clientes de negócios esperam estar continuamente on-line durante o horário comercial. Tanto pequenas empresas, como as agências de viagens com três funcionários, quanto as grandes corporações têm vários computadores conectados por uma LAN. Alguns computadores são PCs de funcionários; outros podem ser servidores da Web. Em geral, existe um roteador na LAN que está conectada ao ISP por uma linha dedicada com a finalidade de fornecer conectividade contínua. Essa organização significa que cada computador deve ter seu próprio endereço IP durante o dia inteiro. Na realidade, o número total de computadores pertencentes a todos os clientes comerciais combinados não pode ultrapassar o número de endereços IP que o ISP tem. No caso de um endereço /16, isso limita o número total de computadores a 65.534. Para um ISP com dezenas de milhares de clientes comerciais, esse limite será ultrapassado rapidamente.

Para piorar, mais e mais usuários estão assinando os serviços de ADSL ou Internet via cabo. Duas características desses serviços são (1) o usuário recebe um endereço IP permanente e (2) não existe nenhuma tarifa por conexão (apenas uma tarifa mensal), de forma que muitos usuários de ADSL e cabo simplesmente ficam conectados de modo permanente. Esse desenvolvimento acelera a redução da quantidade de endereços IP. Atribuir endereços IP no momento da utilização,

como ocorre no caso dos usuários de discagem, não tem utilidade, porque o número de endereços IP em uso em qualquer instante pode ser muitas vezes maior que o número de clientes do ISP.

Apenas para complicar um pouco mais, muitos usuários de ADSL e cabo têm dois ou mais computadores em casa, muitas vezes um computador para cada membro da família, e todos eles querem estar on-line o tempo todo, usando o único endereço IP que o ISP lhes forneceu. A solução aqui é conectar todos os PCs por meio de uma LAN e inserir um roteador nessa LAN. Do ponto de vista do ISP, agora a família equivale a uma pequena empresa com alguns computadores.

O problema de esgotar os endereços IP não é um problema teórico que pode ocorrer em algum momento no futuro distante. Ele está acontecendo aqui mesmo e agora mesmo. A solução a longo prazo é a Internet inteira migrar para o IPv6, que tem endereços de 128 bits. Essa transição está ocorrendo com lentidão e a conclusão do processo irá demorar muitos anos. Em conseqüência disso, algumas pessoas consideraram necessário fazer uma rápida correção a curto prazo. Essa correção veio sob a forma da **NAT (Network Address Translation)**, descrita na RFC 3022 e que resumiremos a seguir. Para obter informações adicionais, consulte (Dutcher, 2001).

A idéia básica por trás da NAT é atribuir a cada empresa um único endereço IP (ou no máximo, um número pequeno deles) para tráfego da Internet. *Dentro* da empresa, todo computador obtém um endereço IP exclusivo, usado para roteamento do tráfego interno. Porém, quando um pacote sai da empresa e vai para o ISP, ocorre uma conversão de endereço. Para tornar esse esquema possível, três intervalos de endereços IP foram declarados como privativos. As empresas podem utilizá-los internamente como desejarem. A única regra é que nenhum pacote contendo esses endereços pode aparecer na própria Internet. Os três intervalos reservados são:

10.0.0.0 — 10.255.255.255/8 (16.777.216 hosts)

172.16.0.0 — 172.31.255.255/12 (1.048.576 hosts)

192.168.0.0 — 192.168.255.255/16 (65.536 hosts)

O primeiro intervalo permite a utilização de 16.777.216 endereços (com exceção de 0 e –1, como sempre) e é a escolha habitual da maioria das empresas, mesmo que elas não necessitem de tantos endereços.

A operação da NAT é mostrada na Figura 5.60. Dentro das instalações da empresa, toda máquina tem um endereço exclusivo da forma 10.*x*.*y*.*z*. Porém, quando um pacote deixa as instalações da empresa, ele passa por uma **caixa NAT** que converte o endereço de origem IP interno, 10.0.0.1 na figura, no endereço IP verdadeiro da empresa, 198.60.42.12 nesse exemplo. Com freqüência, a caixa NAT é combinada em um único dispositivo com um firewall, que oferece segurança por meio do controle cuidadoso do que entra na empresa e do que sai dela. Estudaremos os firewalls no Capítulo 8. Também é possível integrar a caixa NAT ao roteador da empresa.

[arte: ver original p. 446]

[Dísticos]

[1] LAN da empresa

[2] Pacote antes da conversão

[3] Pacote após a conversão

[4] Roteador da empresa

[5] PC

[6] Caixa NAT/firewall

[7] Linha dedicada

[8] Roteador do ISP

[9] Servidor

[10] Limite das instalações da empresa

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.60

[FL] Posicionamento e operação de uma caixa NAT

Até agora, deixamos de lado um pequenino detalhe: quando a resposta volta (por exemplo, de um servidor da Web), ela é naturalmente endereçada para 198.60.42.12; então, como a caixa NAT sabe por qual endereço deve substituir o endereço da resposta? Aqui reside o problema com a NAT. Se houvesse um campo sobressalente no cabeçalho IP, esse campo poderia ser usado para controlar qual foi o transmissor real, mas só resta um bit ainda não utilizado. Em princípio, uma nova opção poderia ser criada para conter o endereço de origem verdadeiro, mas isso exigiria mudar o código do IP em todas as máquinas na Internet inteira para manipular a nova opção. Essa não é uma alternativa promissora para uma correção rápida.

O que realmente aconteceu é descrito a seguir. Os projetistas da NAT observaram que a maioria dos pacotes IP transporta uma carga útil TCP ou UDP. Quando estudarmos o TCP e o UDP no Capítulo 6, veremos que ambos têm cabeçalhos contendo uma porta de origem e uma porta de destino. Descreveremos a seguir apenas as portas TCP, mas o mesmo princípio é válido para as portas UDP. As portas são inteiros de 16 bits que indicam onde a conexão TCP começa e termina. Essas portas fornecem o campo necessário para fazer a NAT funcionar.

Quando um processo deseja estabelecer uma conexão TCP com um processo remoto, ele se associa a uma porta TCP não utilizada em sua própria máquina. Essa porta é chamada **porta de origem** e informa ao código do TCP para onde devem ser enviados os pacotes que chegarem pertencentes a essa conexão. O

processo também fornece uma **porta de destino** para informar a quem devem ser entregues os pacotes no lado remoto. As portas de 0 a 1023 são reservadas para serviços conhecidos. Por exemplo, a porta 80 é usada por servidores da Web, de forma que clientes remotos possam localizá-los. Cada mensagem TCP enviada contém uma porta de origem e uma porta de destino. Juntas, essas portas servem para identificar os processos que utilizam a conexão em ambas as extremidades.

Uma analogia deve tornar mais claro o uso das portas. Imagine uma empresa com um único número de telefone principal. Quando as pessoas ligam para o número principal, acessam um operador que pergunta qual extensão elas desejam, e em seguida as conecta a essa extensão. O número principal é análogo ao endereço IP da empresa, e as extensões em ambas as extremidades são análogas às portas. As portas constituem um grupo extra de 16 bits de endereçamento que identificam o processo que receberá cada pacote de entrada.

Usando o campo *Source port*, podemos resolver nosso problema de mapeamento. Sempre que um pacote de saída entra na caixa NAT, o endereço de origem 10.*x.y.z* é substituído pelo endereço IP verdadeiro da empresa. Além disso, o campo *Source port* do TCP é substituído por um índice para a tabela de conversão de 65.536 entradas da caixa NAT. Essa entrada de tabela contém a porta de origem e o endereço IP original. Por fim, tanto o total de verificação do cabeçalho IP quanto do cabeçalho TCP são recalculados e inseridos no pacote. É necessário substituir o campo *Source port*, porque as conexões das máquinas 10.0.0.1 e 10.0.0.2 podem, por exemplo, não usar a porta 5000, e assim o campo *Source port* sozinho não é suficiente para identificar o processo transmissor.

Quando um pacote chega à caixa NAT vindo do ISP, o campo *Source port* do cabeçalho de TCP é extraído e usado como índice para a tabela de mapeamento da caixa NAT. A partir da entrada localizada, o endereço IP interno e o campo *Source port* do TCP original são extraídos e inseridos no pacote. Em seguida, são

recalculados os totais de verificação do IP e do TCP e inseridos no pacote. O pacote é então repassado ao roteador da empresa para entrega normal, utilizando o endereço 10.*x.y.z*.

A NAT também pode ser usada para atenuar a escassez de endereços IP para usuários de ADSL e cabo. Quando o ISP atribui um endereço a cada usuário, ele utiliza endereços 10.*x.y.z*. Quando pacotes de máquinas do usuário saem do ISP e entram na Internet principal, eles passam por uma caixa NAT que faz a conversão de seus endereços no endereço Internet verdadeiro do ISP. Na volta, os pacotes sofrem o mapeamento inverso. Nesse aspecto, para o restante da Internet, o ISP e seus usuários individuais de ADSL/cabo parecem ser apenas uma grande empresa.

Embora esse tipo de esquema resolva o problema, muitas pessoas na comunidade IP o consideram uma abominação. Em resumo, aqui estão algumas objeções.

Primeiro, a NAT viola o modelo arquitetônico do IP, que estabelece que todo endereço IP identifica de forma exclusiva uma única máquina em todo o mundo. Toda a estrutura de software da Internet se baseia nesse fato. Com a NAT, milhares de máquinas podem usar (e usam) o endereço 10.0.0.1.

Em segundo lugar, a NAT faz a Internet mudar suas características de rede sem conexões para uma espécie de rede orientada a conexões. O problema é que a caixa NAT deve manter informações (o mapeamento) para cada conexão que passa por ela. Manter o estado da conexão é uma propriedade das redes orientadas a conexões, e não das redes sem conexões. Se a caixa NAT sofrer uma pane e sua tabela de mapeamento se perder, todas as conexões TCP serão destruídas. Na ausência da NAT, panes em roteadores não terão nenhum efeito sobre o TCP. O processo transmissor simplesmente entrará em timeout dentro de alguns segundos e retransmitirá todos os pacotes não confirmados. Com a NAT, a Internet se torna tão vulnerável quanto uma rede comutada por circuitos.

Em terceiro lugar, a NAT viola a regra mais fundamental da distribuição de protocolos em camadas: a camada *k* não pode fazer quaisquer suposições sobre o que a camada *k* + 1 inseriu no campo de carga útil. Esse princípio básico existe para manter as camadas independentes. Se o TCP for atualizado mais tarde para TCP-2, com um layout de cabeçalho diferente (por exemplo, portas de 32 bits), a NAT falhará. Toda a idéia de protocolos em camadas tem o objetivo de assegurar que as mudanças em uma camada não exigirão mudanças em outras camadas. A NAT destrói essa independência.

Em quarto lugar, os processos na Internet não são obrigados a usar o TCP ou o UDP. Se um usuário na máquina *A* decidir empregar algum novo protocolo de transporte para se comunicar com um usuário na máquina *B* (por exemplo, no caso de uma aplicação de multimídia), a introdução de uma caixa NAT fará a aplicação falhar, porque a caixa NAT não será capaz de localizar corretamente o campo *Source port* do TCP.

Em quinto lugar, algumas aplicações inserem endereços IP no corpo do texto. O receptor então extrai esses endereços e os utiliza. Tendo em vista que a NAT nada sabe sobre esses endereços, ela não pode substituí-los; assim, qualquer tentativa de usá-los no lado remoto falhará. O **FTP (File Transfer Protocol)** padrão funciona dessa maneira e pode falhar na presença da NAT, a menos que sejam adotadas precauções especiais. De modo semelhante, o protocolo de telefonia da Internet H.323 (que estudaremos no Capítulo 7) tem essa propriedade e pode falhar na presença da NAT. Talvez seja possível corrigir a NAT para que ela funcione com o H.323, mas ter de corrigir o código na caixa NAT toda vez que surge uma nova aplicação não é uma boa idéia.

Em sexto lugar, como o campo *Source port* do TCP é de 16 bits, no máximo 65.536 máquinas podem ser mapeadas em um endereço IP. Na realidade, o número é um pouco menor, porque as primeiras 4096 portas estão reservadas

para usos especiais. Porém, se vários endereços IP estiverem disponíveis, cada um deles poderá manipular até 61.440 máquinas.

Esses e outros problemas com a NAT são discutidos na RFC 2993. Em geral, os opositores da NAT afirmam que, solucionar o problema de insuficiência de endereços IP com uma correção temporária e detestável, significa reduzir a pressão para implementar a verdadeira solução, ou seja, a transição para o IPv6, e isso é ruim.

[T3] 5.6.3 Protocolos de controle da Internet

Além do IP, que é utilizado para a transferência de dados, a Internet tem diversos protocolos de controle usados na camada de rede, incluindo ICMP, ARP, RARP, BOOTP e DHCP. Nesta seção, examinaremos cada um deles.

[T4] ICMP (Internet Control Message Protocol)

A operação da Internet é monitorada rigorosamente pelos roteadores. Quando ocorre algo inesperado, o evento é reportado pelo **ICMP (Internet Control Message Protocol)**, que também é usado para testar a Internet. Existe aproximadamente uma dezena de tipos de mensagens ICMP definidos. Os mais importantes estão listados na Figura 5.61. Cada tipo de mensagem ICMP é encapsulado em um pacote IP.

[arte: ver original p. 449]

[T]Tabela

Tipo de mensagem	**Descrição**
Destination unreachable	Não foi possível entregar o pacote
Time exceeded	O campo Time to live chegou a 0
Parameter problem	Campo de cabeçalho inválido
Source quench	Pacote regulador

Redirect Ensina geografia a um roteador

Echo Pergunta a uma máquina se ela está ativa

Echo reply Sim, estou ativa

Timestamp request Igual a Echo, mas com timbre de hora

Timestamp reply Igual a Echo reply, mas com o timbre de hora

[F]Figura 5.61

[FL] Os principais tipos de mensagens ICMP

A mensagem DESTINATION UNREACHABLE é usada quando a sub-rede ou um roteador não consegue localizar o destino, ou quando um pacote com o bit *DF* não pode ser entregue, porque há uma rede de "pacotes pequenos" no caminho.

A mensagem TIME EXCEEDED é enviada quando um pacote é descartado porque seu contador chegou a zero. Esse evento é um sintoma de que os pacotes estão entrando em loop, de que há um enorme congestionamento ou de que estão sendo definidos valores muito baixos para o timer.

A mensagem PARAMETER PROBLEM indica que um valor inválido foi detectado em um campo de cabeçalho. Esse problema indica a existência de um bug no software IP do host transmissor ou, possivelmente, no software de um roteador pelo qual o pacote transitou.

Antes, a mensagem SOURCE QUENCH era usada para ajustar os hosts que estivessem enviando pacotes demais. Quando recebia essa mensagem, um host devia desacelerar sua operação. Essa mensagem raramente é usada hoje porque, quando ocorre um congestionamento, esses pacotes tendem a colocar mais lenha na fogueira. Atualmente, o controle de congestionamento da Internet é feito, em grande parte, na camada de transporte; vamos estudá-lo em detalhes no Capítulo 6.

A mensagem REDIRECT é usada quando um roteador percebe que o pacote pode

ter sido roteado incorretamente. Ela é usada pelo roteador para informar ao host transmissor o provável erro.

As mensagens ECHO e ECHO REPLY são usadas para verificar se um determinado destino está ativo e acessível. Ao receber a mensagem ECHO, o destino deve enviar de volta uma mensagem ECHO REPLY. As mensagens TIMESTAMP REQUEST e TIMESTAMP REPLY são semelhantes, exceto pelo fato de o tempo de chegada da mensagem e o tempo de saída da resposta serem registrados na mensagem de resposta. Esse recurso é usado para medir o desempenho da rede.

Além dessas mensagens, foram definidas outras. A lista on-line é mantida agora em *www.iana.org/assignements/icmp-parameters*.

[T4] ARP (Address Resolution Protocol)

Embora na Internet cada máquina tenha um (ou mais) endereços IP, na verdade, eles não podem ser usados para transmitir pacotes, pois o hardware da camada de enlace de dados não reconhece endereços da Internet. Hoje em dia, muitos hosts de empresas e universidades estão associados a uma LAN por uma placa de interface que só reconhece endereços de LANs. Por exemplo, cada placa Ethernet fabricada é equipada com um endereço Ethernet de 48 bits. Os fabricantes de placas Ethernet solicitam um bloco de endereços de uma autoridade central para assegurar que duas placas não tenham o mesmo endereço (evitando conflitos, caso as duas estejam na mesma LAN). As placas enviam e recebem quadros com base em endereços Ethernet de 48 bits. Elas nada sabem sobre endereços IP de 32 bits.

Agora, surge a seguinte pergunta: De que forma os endereços IP são mapeados nos endereços da camada de enlace de dados, como é o caso dos endereços Ethernet? Para explicar como esse processo funciona, usaremos o exemplo da Figura 5.62, na qual é ilustrada uma pequena universidade com diversas redes da

classe C (agora chamada /24). Aqui, temos duas redes Ethernet, uma no departamento de ciência da computação com o endereço IP 192.31.65.0, e outra no departamento de engenharia elétrica com o endereço IP 192.31.63.0. As duas estão conectadas por um anel de backbone do campus (por exemplo, FDDI) cujo endereço IP é 192.31.60.0. Cada máquina de uma rede Ethernet tem um endereço Ethernet exclusivo, identificado pelos rótulos *E*1 a *E*6, e cada máquina do anel FDDI tem um endereço FDDI, identificado pelos rótulos de *F*1 a *F*3.

Começaremos examinando como um usuário no host 1 envia um pacote para um usuário no host 2. Vamos supor que o transmissor conheça o nome do receptor pretendido, talvez algo como *mary@eagle.cs.uni.edu*. A primeira etapa é encontrar o endereço IP do host 2, conhecido como *eagle.cs.uni.edu*. Essa pesquisa é realizada pelo DNS (Domain Name System), que estudaremos no Capítulo 7. No momento, supomos apenas que o DNS retorna o endereço IP correspondente ao host 2 (192.31.65.5).

Em seguida, o software da camada superior do host 1 constrói um pacote com 192.31.65.5 no campo *Destination address* e o fornece ao software IP para transmissão. O software IP pode examinar o endereço e constatar que o destino está em sua própria rede, mas ele precisa encontrar de alguma forma o endereço Ethernet da máquina de destino. Uma solução é ter um arquivo de configuração em algum lugar no sistema que faça o mapeamento de endereços IP em endereços Ethernet. Embora essa solução sem dúvida seja possível, no caso de organizações com milhares de máquinas, manter todos esses arquivos atualizados é uma tarefa demorada e propensa a erros.

[arte: ver original p. 451]

[Dísticos]

[1] O roteador CC tem dois endereços IP

192.31.60.4

192.31.65.1

[2] Para WAN

[3] O roteador EE tem dois endereços IP

192.31.60.7

192.31.63.3

[4] 192.31.65.7 192.31.65.5

[5] 192.31.63.8

[6] Endereços Ethernet

[7] Ethernet CC

192.31.65.0

[8] Anel FDDI do campus

192.31.60.0

[9] Ethernet EE

192.31.63.0

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.62

[FL] Três redes /24 interconectadas: duas redes Ethernet e um anel FDDI

Uma solução melhor seria o host 1 enviar um pacote de difusão para a Ethernet perguntando: A quem pertence o endereço IP 192.31.65.5? A difusão chegará a cada máquina da Ethernet 192.31.65.0, e cada uma delas verificará seu endereço IP. Somente o host 2 responderá com seu endereço Ethernet (*E*2). Dessa forma, o host 1 descobrirá que o endereço IP 192.31.65.5 está no host que tem o endereço Ethernet *E*2. O protocolo usado para fazer essa pergunta e receber a resposta é chamado **ARP (Address Resolution Protocol)**. Quase todas as máquinas

na Internet executam esse protocolo. Ele é definido na RFC 826.

A vantagem do uso do ARP sobre os arquivos de configuração é a simplicidade. O administrador do sistema não tem muito a fazer, a não ser atribuir um endereço IP a cada máquina e tomar decisões em relação às máscaras de sub-rede. O ARP faz o resto.

Nesse ponto, o software IP do host 1 constrói um quadro Ethernet endereçado a *E*2, coloca o pacote IP (endereçado a 192.31.65.5) no campo de carga útil e o envia à Ethernet. A placa Ethernet do host 2 detecta esse quadro, reconhece-o como um quadro destinado a ela, recolhe-o e causa uma interrupção. O driver Ethernet extrai o pacote IP da carga útil e o repassa ao software IP, que verifica se ele está corretamente endereçado, e depois o processa.

São possíveis várias otimizações para tornar o ARP mais eficiente. Em primeiro lugar, depois que uma máquina executa o ARP, ela armazena o resultado em um cache, caso precise entrar em contato com a mesma máquina pouco depois. Da próxima vez, ela encontrará o mapeamento em seu próprio cache, eliminando assim a necessidade de uma segunda transmissão. Em muitos casos, o host 2 precisará enviar uma resposta, o que também forçará a execução do ARP para determinar o endereço Ethernet do transmissor. Essa difusão do ARP pode ser evitada fazendo-se com que o host 1 inclua seu mapeamento de IP para Ethernet no pacote ARP. Quando a difusão do ARP chega ao host 2, o par (192.31.65.7, E1) é inserido no cache ARP do host 2 para uso futuro. Na realidade, todas as máquinas da rede Ethernet podem inserir esse mapeamento em seus caches ARP.

Uma outra otimização seria fazer com que cada máquina difundisse seu mapeamento ao ser reinicializada. Essa difusão em geral é feita quando um ARP procura seu próprio endereço IP. Não deve haver uma resposta, mas um efeito colateral da difusão é criar uma entrada no cache ARP de todas as máquinas. Se chegar uma resposta (inesperada), isso significa que o mesmo endereço IP foi

atribuído a duas máquinas. A nova máquina deve informar esse fato ao administrador do sistema e não poderá ser reinicializada.

Para permitir alterações no mapeamento quando, por exemplo, uma placa Ethernet é danificada e precisa ser substituída por uma nova (e portanto por outro endereço Ethernet), as entradas no cache ARP sofrem um timeout após alguns minutos.

Vamos examinar novamente a Figura 5.62; a diferença é que agora o host 1 deseja enviar um pacote ao host 4 (192.31.63.8). O uso do ARP apresentará problemas, porque o host 4 não verá a difusão (os roteadores não encaminham difusões no nível da rede Ethernet). Existem duas soluções. Na primeira delas, o roteador CC poderia ser configurado para responder a solicitações ARP para a rede 192.31.63.0 (e talvez para outras redes locais). Nesse caso, o host 1 criará uma entrada de cache ARP (192.31.63.8, E3) e enviará todo o tráfego destinado ao host 4 para o roteador local. Essa solução é chamada **ARP proxy**. A segunda solução é fazer com que o host 1 veja imediatamente que o destino está em uma rede remota e envie todo o tráfego para um endereço Ethernet padrão que trate do tráfego remoto, nesse caso *E*3. Essa solução não requer que o roteador CC saiba a que redes remotas está servindo.

De qualquer forma, o que acontece é que o host 1 coloca o pacote IP no campo de carga útil de um quadro Ethernet endereçado a *E*3. Quando obtém o quadro Ethernet, o roteador CC remove o pacote IP do campo de carga útil e procura o endereço IP nas tabelas de roteamento. Além disso, esse roteador descobre que os pacotes destinados à rede 192.31.63.0 devem ir para o roteador 192.31.60.7. Se ainda não souber o endereço FDDI de 192.31.60.7, o roteador transmitirá um pacote ARP para o anel e descobrirá que seu endereço de anel é *F*3. Em seguida, ele incluirá o pacote no campo de carga útil de um quadro FDDI endereçado a *F*3 e o colocará no anel.

No roteador EE, o driver FDDI remove o pacote do campo de carga útil e o envia ao software IP, que constata a necessidade de enviar o pacote para 192.31.63.8. Se esse endereço IP não estiver em seu cache ARP, o software transmitirá uma solicitação ARP através da rede Ethernet do EE e descobrirá que o endereço de destino é *E*6. Em seguida, o software montará um quadro Ethernet endereçado a *E*6, colocará o pacote no campo de carga útil e fará sua transmissão na Ethernet. Quando o quadro Ethernet chegar ao host 4, o pacote será extraído do quadro e repassado ao software IP para processamento.

Ir do host 1 até uma rede distante passando por uma WAN é um processo praticamente igual, exceto pelo fato de, dessa vez, as tabelas do roteador CC o informarem de que ele deve usar o roteador da WAN, cujo endereço FDDI é *F*2.

[T4] RARP, BOOTP e DHCP

O ARP resolve o problema de encontrar um endereço Ethernet que corresponda a um determinado endereço IP. Às vezes, é necessário resolver o problema inverso: Qual é o endereço IP correspondente a um endereço Ethernet? Isso ocorre especificamente quando uma estação de trabalho sem disco é inicializada. Em geral, essa máquina obterá a imagem binária de seu sistema operacional a partir de um servidor de arquivos remoto. No entanto, como ela descobrirá seu endereço IP?

A primeira solução imaginada foi usar o **RARP (Reverse Address Resolution Protocol)** (definido na RFC 903). Esse protocolo permite que uma estação de trabalho recém-inicializada transmita seu endereço Ethernet e informe: Meu endereço Ethernet de 48 bits é 14.04.05.18.01.25. Alguém conhece meu endereço IP? O servidor RARP vê essa solicitação, procura o endereço Ethernet em seus arquivos de configuração e envia de volta o endereço IP correspondente.

O uso do RARP é melhor que a inclusão de um endereço IP na imagem de

memória, porque permite que a mesma imagem seja usada em todas as máquinas. Se o endereço IP fosse embutido na imagem, cada estação de trabalho precisaria ter sua própria imagem.

Uma desvantagem do RARP é que ele utiliza um endereço de destino composto somente por valores 1 (difusão limitada) para chegar ao servidor RARP. Entretanto, essas difusões não são encaminhadas pelos roteadores; portanto, é necessário um servidor RARP em cada rede. Para resolver esse problema, foi criado um protocolo de inicialização alternativo, chamado **BOOTP.** Diferente do RARP, o BOOTP utiliza mensagens UDP, que são encaminhadas pelos roteadores. O BOOTP também fornece informações adicionais a uma estação de trabalho sem disco, inclusive o endereço IP do servidor de arquivos que contém a imagem de memória, o endereço IP do roteador padrão e a máscara de sub-rede a ser usada. O BOOTP é descrito nas RFCs 951, 1048 e 1084.

Um problema sério com o BOOTP é que ele exige configuração manual de tabelas que mapeiam endereços IP para endereços Ethernet. Quando um novo host é adicionado a uma LAN, ele não pode usar o BOOTP enquanto um administrador não tiver atribuído a ele um endereço IP e inserido manualmente seu par (endereço Ethernet, endereço IP) nas tabelas de configuração do BOOTP. Para eliminar essa etapa propensa a erros, o BOOTP foi ampliado e recebeu um novo nome, **DHCP (Dynamic Host Configuration Protocol).** O DHCP permite a atribuição manual e a atribuição automática de endereços IP. Ele é descrito nas RFCs 2131 e 2132. Na maioria dos sistemas, o DHCP substituiu em grande parte o RARP e o BOOTP.

Como o RARP e o BOOTP, o DHCP se baseia na idéia de um servidor especial que atribui endereços IP a hosts que solicitam um endereço. Esse servidor não precisa estar na mesma LAN em que se encontra o host solicitante. Tendo em vista que o servidor DHCP pode não estar acessível por difusão, um **agente de retransmissão**

DHCP é necessário em cada LAN, como mostra a Figura 5.63.

Para encontrar seu endereço IP, uma máquina recém-inicializada transmite por difusão um pacote DHCP DISCOVER. O agente de retransmissão DHCP em sua LAN intercepta todas as difusões do DHCP. Ao encontrar um pacote DHCP DISCOVER, ele envia o pacote como um pacote de unidifusão ao servidor DHCP, talvez em uma rede distante. O único item de informações que o agente de retransmissão precisa ter é o endereço IP do servidor DHCP.

[arte: ver original p. 454]

[Dísticos]

[1]Host recém-inicializado, procurando por seu endereço IP

[2] Retransmissão DHCP

[3] Outras redes

[4] Roteador Servidor DHCP

[5] Pacote DHCP Discover (difusão)

[6] Pacote de unidifusão do agente de retransmissão DHCP para o servidor DHCP

[F]Figura 5.63

[FL] Operação do DHCP

Uma questão que surge com a atribuição automática de endereços IP de um pool é o tempo durante o qual um endereço IP deve ser alocado. Se um host deixar a rede e não retornar seu endereço IP ao servidor DHCP, esse endereço será permanentemente perdido. Depois de um certo período, muitos endereços poderão se perder. Para evitar que isso aconteça, a atribuição de endereços IP pode se referir a um período fixo, uma técnica chamada **arrendamento (leasing).** Pouco antes de expirar o prazo de arrendamento, o host deve solicitar ao DHCP uma renovação. Se ele deixar de fazer uma solicitação ou se a solicitação for negada, o host não poderá mais usar o endereço IP que recebeu antes.

[T3] 5.6.4 OSPF — Interior Gateway Routing Protocol

Concluímos nosso estudo dos protocolos de controle da Internet, e agora chegou o momento de passarmos ao tópico seguinte: o roteamento na Internet. Como mencionamos antes, a Internet é formada por um grande número de sistemas autônomos (SAs). Cada SA é operado por uma organização diferente e pode usar seu próprio algoritmo de roteamento interno. Por exemplo, as redes internas das empresas *X*, *Y* e *Z* em geral serão vistas como três SAs, se todas estiverem na Internet. Internamente, todas três podem usar algoritmos de roteamento específicos. Apesar disso, o fato de haver padrões, mesmo para roteamento interno, simplifica a implementação de fronteiras entre os SAs e permite a reutilização do código. Nesta seção, estudaremos o roteamento em um SA. Na próxima, examinaremos o roteamento entre SAs. Um algoritmo de roteamento em um SA é chamado **protocolo de gateway interior**; um algoritmo para roteamento entre SAs é chamado **protocolo de gateway exterior**.

O protocolo de gateway interior da Internet original era um protocolo de vetor de distância (RIP) baseado no algoritmo de Bellman-Ford, herdado da ARPANET. Ele funcionava bem em sistemas pequenos; no entanto, tudo mudava à medida que os SAs se tornavam maiores. O protocolo também sofria do problema da contagem até infinito e, em geral, de uma convergência lenta; portanto, em maio de 1979 foi substituído por um protocolo de estado de enlace. Em 1988, a Internet Engineering Task Force começou a trabalhar em um sucessor, chamado de **OSPF (Open Shortest Path First)**, que se tornou um padrão em 1990. Muitos fornecedores de roteadores passaram a aceitá-lo, e ele se tornou o principal protocolo de gateway interior. A seguir, faremos um esboço de como funciona o OSPF. Para obter informações mais completas, consulte a RFC 2328.

Devido à grande experiência com outros protocolos de roteamento, o grupo que

projetou o novo protocolo tinha uma longa lista de requisitos que deveriam ser atendidos. Primeiro, o algoritmo teria de ser amplamente divulgado na literatura especializada, daí o "O" (de Open, ou aberto) da sigla OSPF. Uma solução patenteada de uma única empresa não funcionaria nesse caso. Em segundo lugar, o novo protocolo teria de admitir uma variedade de unidades de medida de distância, inclusive a distância física, o retardo etc. Em terceiro lugar, ele teria de ser um algoritmo dinâmico, que se adaptasse de forma rápida e automática a alterações na topologia.

Em quarto lugar, uma novidade no caso do OSPF: ele tinha de admitir o roteamento baseado no tipo de serviço. O novo protocolo deveria ser capaz de rotear tráfego de tempo real de uma determinada maneira e outro tipo de tráfego de maneira diferente. O protocolo IP tem um campo *Type of service*, mas nenhum protocolo de roteamento existente o utilizava. Esse campo foi incluído no OSPF, mas ninguém o utilizava ainda e, mais tarde, ele foi removido.

Como um quinto requisito, relacionado aos anteriores, o novo protocolo tinha de balancear a carga, dividindo-a por várias linhas. A maioria dos protocolos anteriores enviava todos os pacotes pela melhor rota. A segunda melhor rota não era usada. Em muitos casos, a divisão da carga por várias linhas proporciona melhor desempenho.

Em sexto lugar, era necessário o suporte para sistemas hierárquicos. Em 1988, a Internet tinha crescido tanto que nenhum roteador era capaz de conhecer a topologia inteira. O novo protocolo de roteamento teve de ser projetado de forma que nenhum roteador fosse obrigado a conhecer a topologia.

Em sétimo lugar, era necessário um certo nível de segurança para evitar que estudantes em busca de diversão tentassem enganar os roteadores, enviando-lhes falsas informações de roteamento. Por fim, era necessário tomar alguma providência para lidar com os roteadores conectados à Internet por meio de um

túnel — assunto que os protocolos anteriores não dominavam muito bem.

O OSPF é compatível com três tipos de conexões e redes:

1. Linhas ponto a ponto entre, exatamente, dois roteadores.

2. Redes de multiacesso com difusão (por exemplo, a maioria das LANs).

3. Redes de multiacesso sem difusão (por exemplo, a maioria das WANs comutadas por pacotes).

Uma rede de **multiacesso** é aquela que pode ter vários roteadores e cada um dos quais pode se comunicar diretamente com todos os outros. Todas as LANs e WANs têm essa propriedade. A Figura 5.64(a) mostra um SA contendo todos os três tipos de redes. Observe que os hosts nem sempre têm uma função no OSPF. O OSPF funciona transformando o conjunto de redes, roteadores e linhas reais em um grafo orientado, no qual se atribui um custo (distância, retardo etc.) a cada arco. Em seguida, o OSPF calcula o caminho mais curto com base nos pesos dos arcos. Uma conexão serial entre dois roteadores é representada por um par de arcos, um em cada sentido. Seus pesos podem ser diferentes. Uma rede de multiacesso é representada por um nó para a própria rede e por um nó para cada roteador. Os arcos entre o nó da rede e os roteadores têm peso 0 e foram omitidos do grafo.

[arte: ver original p. 456]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 5.64

[FL] (a) Um sistema autônomo. (b) Uma representação gráfica de (a)

A Figura 5.64(b) mostra a representação gráfica da rede da Figura 5.64(a). Os pesos são simétricos, a menos que esteja indicado algo em contrário. O que o

OSPF faz fundamentalmente é representar a rede real como um grafo e, em seguida, calcular o caminho mais curto de cada roteador para cada outro roteador.

Muitos dos SAs da Internet são grandes e difíceis de gerenciar. O OSPF permite que eles sejam divididos em **áreas** numeradas; uma área é uma rede ou um conjunto de redes contíguas. Essas áreas não se sobrepõem, mas não precisam ser completas, ou seja, talvez alguns roteadores não pertençam a nenhuma área. Uma área é uma generalização de uma sub-rede. Fora de uma área, a topologia e os detalhes da sub-rede não são visíveis.

Cada SA possui uma área de **backbone**, chamada área 0. Todas as áreas estão conectadas ao backbone, possivelmente por túneis, permitindo que se vá de uma área do SA para qualquer outra via backbone. Um túnel é representado no grafo como um arco e tem um custo específico. Cada roteador conectado a duas ou mais áreas faz parte do backbone. Como em outras áreas, a topologia do backbone não pode ser vista fora dele.

Em uma área, cada roteador tem o mesmo banco de dados de estado de enlace e utiliza o mesmo algoritmo de caminho mais curto. Sua principal função é calcular o caminho mais curto entre ele e os outros roteadores da área, incluindo o roteador conectado ao backbone (deve existir pelo menos um roteador). Um roteador que se conecta a duas áreas precisa dos bancos de dados de ambas as áreas e deve utilizar o algoritmo de caminho mais curto em cada uma delas separadamente.

Durante a operação normal, talvez sejam necessários três tipos de rotas: entre áreas, na mesma área e entre sistemas autônomos. As rotas na mesma área são as mais fáceis, pois o roteador de origem já conhece o caminho mais curto para o roteador de destino. O roteamento entre áreas sempre acontece em três etapas: vai da origem para o backbone; atravessa o backbone até a área de destino; vai

até o destino. Esse algoritmo força uma configuração em estrela no OSPF, com o backbone sendo o hub e as outras áreas sendo os raios. Os pacotes são roteados da origem para o destino "no estado em que se encontram". Eles não são encapsulados ou colocados em túneis, a menos que estejam indo para uma área cuja única conexão com o backbone seja um túnel. A Figura 5.65 mostra parte da Internet com SAs e áreas.

O OSPF distingue quatro classes de roteadores:

1. Os roteadores internos, que ficam inteiramente em uma área.
2. Os roteadores de borda de área, que conectam duas ou mais áreas.
3. Os roteadores de backbone, que ficam no backbone.
4. Os roteadores de fronteira do SA, que interagem com roteadores de outros SAs.

Essas classes podem se sobrepor. Por exemplo, todos os roteadores de borda fazem parte do backbone automaticamente. Além disso, um roteador que esteja no backbone, mas que não faça parte de nenhuma outra área, também é um roteador interno. Exemplos de todas as quatro classes de roteadores são ilustrados na Figura 5.65.

Quando é inicializado, um roteador envia mensagens HELLO por todas as suas linhas ponto a ponto, transmitindo-as por difusão nas LANs até o grupo que consiste em todos os outros roteadores. Nas WANs, o roteador necessita de algumas informações de configuração para saber quem contactar. A partir das respostas, cada roteador descobre quem são seus vizinhos. Os roteadores da mesma LAN são todos vizinhos.

O OSPF troca informações entre roteadores adjacentes; essas informações não são as mesmas informações trocadas entre os roteadores vizinhos. Em particular, não é útil fazer cada roteador de uma LAN se comunicar com todos os outros roteadores da mesma LAN. Para evitar essa situação, um roteador é eleito como o

roteador designado. Ele é considerado **adjacente** a todos os outros roteadores em sua LAN e troca informações com eles. Roteadores vizinhos que não são adjacentes não trocam informações entre si. Um roteador designado de reserva é sempre mantido atualizado, a fim de facilitar a transição, caso o roteador designado principal venha a falhar e precise ser substituído de imediato.

[arte: ver original p. 458]

[Dísticos]

[1]Roteador de fronteira de SA

[2] Backbone

[3] Roteador de backbone

[4] Área

[5] Roteador de borda de área

[6] O protocolo BGP conecta os SAs

[7] Roteador interno

[8] SA 3 SA 4

[9] SA 1 SA 2

[F]Figura 5.65

[FL] A relação entre SAs, backbones e áreas no OSPF

Durante a operação normal, cada roteador emite periodicamente por inundação mensagens LINK STATE UPDATE para cada um de seus roteadores adjacentes. Essa mensagem informa seu estado e fornece os custos usados no banco de dados da topologia. As mensagens enviadas são confirmadas, a fim de torná-las confiáveis. Cada mensagem tem um número de seqüência, e assim o roteador pode ver se uma mensagem LINK STATE UPDATE recebida é mais antiga ou mais recente do que a atual. Os roteadores também enviam essas mensagens quando uma linha é ativada ou desativada, ou quando seus custos se alteram.

As mensagens DATABASE DESCRIPTION fornecem os números de seqüência de todas as entradas de estado de enlace mantidas no momento pelo transmissor. Comparando seus próprios valores com os valores do transmissor, o receptor pode determinar quem tem os valores mais recentes. Essas mensagens são usadas quando uma linha é interrompida.

Cada parceiro pode solicitar informações de estado de enlace um ao outro, usando mensagens LINK STATE REQUEST. O resultado desse algoritmo é que cada par de roteadores adjacentes verifica quem tem os dados mais recentes, e as novas informações são divulgadas por toda a área. Todas essas mensagens são enviadas como pacotes IP puros. Os cinco tipos de mensagens estão resumidos na Figura 5.66.

[arte: ver original p. 459]

[T]Tabela

Tipo da mensagem	**Descrição**
Hello	Usada para descobrir quem são os vizinhos
Link state update	Fornece os custos do transmissor a seus vizinhos
Link state ack	Confirma a atualização do estado do enlace
Database description	Anuncia quais são as atualizações do transmissor
Link state request	Solicita informações do parceiro

[F]Figura 5.66

[FL] Os cinco tipos de mensagens OSPF

Por fim, podemos juntar todos os fragmentos. Usando o processo de inundação, cada roteador informa todos os outros roteadores de sua área sobre seus vizinhos e custos. Essas informações permitem que cada roteador construa o grafo para a(s) sua(s) área(s) e calcule o caminho mais curto. A área do backbone faz o mesmo. Além disso, os roteadores do backbone aceitam as informações dos

roteadores de borda de área para calcular a melhor rota entre cada roteador do backbone até cada um dos outros roteadores. Essas informações são propagadas para os roteadores de borda de área, que as divulgam em suas áreas. Usando essas informações, um roteador prestes a enviar um pacote entre áreas pode selecionar o melhor roteador de saída para o backbone.

[T3] 5.6.5 BGP — O protocolo de roteamento de gateway exterior

Em um único SA, o protocolo de roteamento recomendado na Internet é o OSPF (embora este não seja o único em uso). Entre SAs é usado outro protocolo, o **BGP (Border Gateway Protocol)**. É necessário um protocolo diferente entre SAs, porque os objetivos de um protocolo de gateway interior e os de um protocolo de gateway exterior não são os mesmos. Tudo o que um protocolo de gateway interior precisa fazer é movimentar pacotes da forma mais eficiente possível, da origem até o destino. Ele não precisa se preocupar com política.

Os roteadores de protocolos de gateway exterior têm de se preocupar muito com política (Metz, 2001). Por exemplo, um SA corporativo talvez precise ter a capacidade de enviar pacotes para qualquer site da Internet e receber pacotes de qualquer site da Internet. Entretanto, é possível que ele não esteja disposto a transportar pacotes que tenham origem em um SA externo e destino em outro SA externo, mesmo que seu próprio SA esteja no caminho mais curto entre os dois SAs externos ("Isso é problema deles, não nosso"). Por outro lado, talvez ele queira transportar pacotes para seus vizinhos ou mesmo para outros SAs específicos, que tenham pago por esse serviço. Por exemplo, as companhias telefônicas talvez fiquem felizes por funcionarem como concessionárias de comunicações para seus clientes, mas não para os outros. Os protocolos de gateway exterior em geral e o BGP em particular forma projetados para permitir a imposição de muitos tipos de políticas de roteamento no tráfego entre SAs.

Em geral, as políticas (ou normas) envolvem considerações políticas, econômicas e de segurança. Alguns exemplos de restrições de roteamento são:

1. Nenhum tráfego deve passar por certos SAs.
2. Nunca colocar o Iraque em uma rota que comece no Pentágono.
3. Não usar os Estados Unidos para ir da Colúmbia Britânica até Ontário.
4. Só passar pela Albânia se não houver nenhuma alternativa para chegar ao destino.
5. O tráfego que começar ou terminar na IBM não deve transitar pela Microsoft.

Em geral, as políticas são configuradas manualmente em cada roteador BGP (ou incluídas com a utilização de algum tipo de script). Elas não fazem parte do protocolo em si.

Do ponto de vista de um roteador BGP, o mundo consiste em SAs e nas linhas que os conectam. Dois SAs são considerados conectados se existe uma linha entre roteadores de borda de cada um deles. Devido ao especial interesse do BGP pelo tráfego, as redes são agrupadas em três categorias. Da primeira categoria fazem parte as **redes stub**, que têm somente uma conexão com o grafo BGP. Elas não podem ser usadas para tráfego, porque não há ninguém do outro lado. Em seguida, temos as **redes multiconectadas**, que podem ser usadas para tráfego, a menos que se recusem. Por fim, temos as **redes de trânsito**, tais como backbones, cujo objetivo é tratar pacotes de terceiros, possivelmente com algumas restrições e em geral com a cobrança de alguma tarifa.

Os pares de roteadores BGP se comunicam entre si, estabelecendo conexões TCP. Esse tipo de operação possibilita uma comunicação confiável e oculta todos os detalhes da rede que está sendo utilizada.

O BGP é fundamentalmente um protocolo de vetor de distância, mas é bem diferente da maioria dos outros, como o RIP. Em vez de apenas manter o custo para cada destino, cada roteador BGP tem controle de qual caminho está sendo

usado. Da mesma forma, em vez de fornecer periodicamente a cada vizinho seu custo estimado para cada destino possível, o roteador BGP informa a seus vizinhos o caminho exato que está usando. Como exemplo, considere os roteadores BGP mostrados na Figura 5.67(a). Em particular, considere a tabela de roteamento de *F*. Suponha que seja usado o caminho *FGCD* para se chegar a *D*. Quando lhe fornecem informações de roteamento, os vizinhos de *F* transmitem seus caminhos completos, como mostra a Figura 5.67(b) (para simplificar, somente o destino *D* é mostrado na figura).

Depois que todos os vizinhos enviam seus caminhos, *F* os examina para verificar qual é o melhor. Rapidamente, *F* descarta os caminhos com origem em *I* e *E*, pois eles passam pelo próprio *F*. Resta então optar por usar *B* e *G*. Cada roteador BGP contém um módulo que examina e conta as rotas para um determinado destino, retornando um número que identifica a "distância" até esse destino em relação a cada rota. Qualquer rota que viole uma restrição política recebe automaticamente uma contagem infinita. Em seguida, o roteador adota a rota com a distância mais curta. A função de contagem não faz parte do protocolo BGP e pode ser qualquer função que os administradores do sistema desejarem.

[arte: ver original p. 461]

[Dísticos]

[1]Informações sobre D que F recebe de seus vizinhos

[2] De B: "Eu utilizo BCD"

De G: "Eu utilizo GCD"

De I: "Eu utilizo IFGCD"

De E: "Eu utilizo EFGCD"

(b)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem

seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 5.67

[FL] (a) Um conjunto de roteadores BGP. (b) Informações enviadas para *F*

O BGP resolve facilmente o problema da contagem até infinito, que atinge outros algoritmos de roteamento com vetor de distância. Por exemplo, suponha que *G* falhe ou que a linha *FG* seja desativada. Então, *F* receberá rotas de seus três vizinhos restantes. Essas rotas são *BCD*, *IFGCD* e *EFGCD*. Imediatamente, ele pode constatar que as duas últimas rotas são inúteis, pois passam através do próprio *F*, e acaba escolhendo *FBCD* como a nova rota. Com freqüência, os outros algoritmos de vetor de distância escolhem a opção errada, porque não são capazes de informar quais dos vizinhos têm rotas independentes para o destino, e quais não as têm. A definição do BGP está nas RFCs 1171 a 1174.

[T3] 5.6.6 Multidifusão na Internet

Em geral, as comunicações IP são feitas entre um transmissor e um receptor. Entretanto, para algumas aplicações, é interessante que um processo seja capaz de transmitir dados para um grande número de receptores simultaneamente. Dentre os exemplos dessa estratégia estão a atualização de bancos de dados distribuídos replicados, a transmissão das cotações de ações para vários corretores e o controle de chamadas de teleconferência digital.

O IP aceita a multidifusão, usando endereços da classe D que identificam um grupo de hosts. Estão disponíveis 28 bits para identificar grupos; portanto, pode haver mais de 250 milhões de grupos ao mesmo tempo. Quando um processo envia um pacote para um endereço da classe D, é feita uma tentativa de entregá-lo a todos os membros do grupo endereçado, mas não há qualquer garantia de que isso realmente acontecerá. É provável que alguns membros não obtenham o

pacote.

São aceitos dois tipos de endereços de grupo: endereços permanentes e endereços temporários. Um grupo permanente está sempre presente e não precisa ser configurado. Cada grupo permanente tem um endereço de grupo permanente. Alguns exemplos de endereços de grupos permanentes são:

224.0.0.1 Todos os sistemas de uma LAN

224.0.0.2 Todos os roteadores de uma LAN

224.0.0.5 Todos os roteadores OSPF de uma LAN

224.0.0.6 Todos os roteadores OSPF designados de uma LAN

Para usar um grupo temporário, primeiro você deve criá-lo. Um processo pode solicitar que seu host se conecte a um grupo específico, e também pode solicitar que seu host saia do grupo. Quando o último processo de um host deixa um grupo, o grupo passa a não mais existir no host. Cada host controla os grupos a que pertencem seus processos atuais.

A multidifusão é implementada por roteadores de multidifusão especiais, que podem ou não ser colocados com os roteadores padrão. Cerca de uma vez por minuto, cada roteador de multidifusão envia um processo de hardware de multidifusão (ou seja, da camada de enlace de dados) para os hosts de sua LAN (endereço 224.0.0.1) solicitando que eles informem os grupos aos quais seus processos pertencem atualmente. Cada host envia respostas a todos os endereços da classe D nos quais está interessado.

Esses pacotes de consulta e resposta utilizam um protocolo chamado **IGMP (Internet Group Management Protocol)**, que é um pouco semelhante ao ICMP. Ele tem apenas dois tipos de pacotes: consulta e resposta, cada um com um formato fixo simples, contendo algumas informações de controle na primeira palavra do campo de carga útil, e um endereço da classe D na segunda palavra. O IGMP é descrito na RFC 1112.

O roteamento por multidifusão é feito com base no uso de árvores de amplitude. Cada roteador de multidifusão troca informações com seus vizinhos usando um protocolo de vetor de distância modificado. Dessa forma, cada um dos vizinhos é capaz de construir uma árvore de amplitude que abrange todos os membros do grupo. Várias otimizações são usadas para podar a árvore, a fim de eliminar roteadores e redes que não interessam a determinados grupos. O protocolo faz uso intenso do recurso de tunneling para não incomodar os nós que se encontram fora da árvore de amplitude.

[T3] 5.6.7 IP móvel

Muitos usuários da Internet têm computadores portáteis e desejam permanecer conectados à Internet quando visitam um site distante e mesmo quando estão em trânsito. Infelizmente, o sistema de endereçamento IP torna muito difícil o trabalho longe de casa. Nesta seção, examinaremos o problema e a solução. Uma descrição mais detalhada é apresentada em (Perkins, 1998a).

O grande vilão é o próprio esquema de endereçamento. Cada endereço IP contém um número de rede e um número de host. Por exemplo, considere a máquina com o endereço IP 160.80.40.20/16. A parte 160.80 fornece o número da rede (8272 em decimal); o número 40.20 é o número do host (10260 em decimal). Os roteadores existentes em todo o mundo têm tabelas de roteamento que informam qual linha deve ser usada para se chegar à rede 160.80. Sempre que um pacote chega a um endereço IP de destino com o formato 160.80.xxx.yyy, ele é enviado por essa linha.

Se subitamente a máquina que tem esse endereço for removida para um site distante, os pacotes destinados a ela continuarão a ser roteados para sua LAN de origem (ou roteador). O dono da máquina não receberá mais mensagens de correio eletrônico, entre outros itens. O fornecimento de um novo endereço IP

para a máquina correspondente à sua nova localização talvez não seja muito interessante, pois muitas pessoas, programas e bancos de dados teriam de ser informados a respeito da alteração.

Outra estratégia seria fazer com que os roteadores utilizassem endereços IP completos para roteamento, em vez de apenas a rede. Entretanto, isso poderia exigir que cada roteador tivesse milhões de entradas nas tabelas, a um custo astronômico para a Internet.

Quando as pessoas começaram a levantar a possibilidade de utilização de hosts móveis, a IETF estabeleceu um grupo de trabalho para encontrar uma solução. Esse grupo de trabalho formulou inúmeros objetivos considerados desejáveis, independente da solução adotada. Os principais eram:

1. Cada host móvel deveria ser capaz de usar seu endereço IP de origem em qualquer lugar.
2. Não eram permitidas alterações de software nos hosts fixos.
3. Não eram permitidas alterações no software e nas tabelas do roteador.
4. Muitos pacotes destinados a hosts móveis não deveriam fazer desvios durante o percurso.
5. Não deveria haver overhead quando um host móvel estivesse em sua origem.

Descrevemos na Seção 5.2.8 a solução escolhida. Em resumo, um site disposto a permitir que seus usuários vaguem sem destino precisa criar um agente interno. Quando um host móvel aparece em um site externo, ele entra em contato com o host externo e o registra. Em seguida, o host externo entra em contato com o agente interno do usuário e lhe fornece um **endereço de tratamento**, que normalmente é o endereço IP do próprio agente externo.

Quando chega à LAN de origem do usuário, o pacote entra em algum roteador conectado a essa LAN. Em seguida, o roteador tenta localizar o host da forma habitual, por exemplo transmitindo por difusão um pacote ARP que pergunta:

Qual é o endereço Ethernet de 160.80.40.20? O agente interno responde a essa consulta fornecendo seu próprio endereço Ethernet. Em seguida, o roteador envia os pacotes destinados a 160.80.40.20 para o agente interno. Por sua vez, o agente interno transporta os pacotes por um túnel até o endereço de tratamento, encapsulando-os no campo de carga útil de um pacote IP endereçado ao agente externo. Depois, o agente externo retira os pacotes do campo de carga útil e os entrega ao endereço de enlace de dados do host móvel. Além disso, o agente interno fornece o endereço de tratamento ao transmissor, de forma que os futuros pacotes possam ser transportados pelo túnel diretamente para o agente externo. Essa solução corresponde a todos os requisitos citados anteriormente.
Vale a pena mencionar um pequeno detalhe. No momento em que o host móvel se movimenta, provavelmente o roteador tem seu endereço Ethernet (prestes a se tornar inválido) armazenado em um cache. Para substituir esse endereço Ethernet pelo endereço do agente interno, é usado um artifício chamado **ARP gratuito**. Trata-se de uma mensagem especial para o roteador, não solicitada e que provoca a substituição de uma determinada entrada de cache; nesse caso, a entrada correspondente ao host móvel prestes a sair. Quando o host móvel retornar mais tarde, será usado o mesmo artifício para atualizar novamente o cache do roteador.
Não há nada no projeto que impeça o host móvel de ser seu próprio agente externo, mas essa estratégia só funcionará se o host móvel (atuando como agente externo) estiver logicamente conectado à Internet em seu site atual. Além disso, o host móvel deve ser capaz de adquirir um endereço IP de tratamento (temporário) para uso. Esse endereço IP deve pertencer à LAN a que ele está conectado no momento.
A solução da IETF para hosts móveis resolve uma série de outros problemas ainda não mencionados. Por exemplo, como os agentes são localizados? A solução é

fazer com que cada agente transmita periodicamente seu endereço e o tipo de serviço que pretende fornecer (por exemplo, interno, externo ou ambos). Quando um host móvel chega a algum lugar, ele só precisa ouvir essas transmissões, denominadas **anúncios**. Como alternativa, o host pode transmitir um pacote anunciando sua chegada e esperar que o agente externo local responda.

Outro problema que precisava ser resolvido era o que fazer no caso de hosts móveis mal-educados que vão embora sem se despedir. A solução seria tornar o registro válido somente durante um intervalo de tempo fixo. Se não for renovado periodicamente, o registro sofrerá um timeout, permitindo que o host externo esvazie suas tabelas.

Outro aspecto é a segurança. Quando um agente interno recebe uma mensagem solicitando que ele encaminhe todos os pacotes destinados a Roberta para algum endereço IP, o melhor é que ele não obedeça, a menos que ele esteja convencido de que Roberta seja a origem dessa solicitação, e não alguém tentando se passar por ela. Os protocolos de autenticação criptográfica são usados com essa finalidade. Estudaremos esses protocolos no Capítulo 8.

Um último ponto abordado pelo grupo de trabalho se relaciona aos níveis de mobilidade. Imagine um avião com uma Ethernet a bordo usada pelos computadores de navegação. Nessa Ethernet, existe um roteador padrão que se comunica com a Internet, instalada em terra, por meio de um enlace de rádio. Um belo dia, um executivo de marketing tem a idéia de instalar conectores Ethernet em todos os braços das poltronas, permitindo que os passageiros com computadores móveis também se conectem.

Agora, temos dois níveis de mobilidade: os computadores da própria aeronave, que são fixos em relação à Ethernet, e os computadores dos passageiros, que são móveis em relação a ela. Além disso, o roteador que está a bordo do avião é móvel em relação aos roteadores instalados em terra. O fato de ser móvel em

relação a um sistema que também é móvel pode ser tratado utilizando-se o tunneling recursivo.

[T3] 5.6.8 IPv6

Embora o CIDR e a NAT ainda tenham alguns anos pela frente, todo mundo percebe que o IP em sua forma atual (IPv4) está com os dias contados. Além desses problemas técnicos, há uma outra questão em paralelo. No início, a Internet era amplamente usada por universidades, indústrias de alta tecnologia e órgãos governamentais dos EUA (especialmente pelo Departamento de Defesa). Com a explosão da Internet a partir de meados da década de 1990, ela começou a ser usada por um grupo diferente de pessoas, em especial pessoas com necessidades específicas. Por um lado, milhares de pessoas com computadores portáteis sem fios a utilizam para estabelecer contato com suas bases. Por outro, com a inevitável convergência das indústrias de informática, comunicação e entretenimento, talvez não demore para que cada telefone e cada televisor do mundo seja um nó da Internet, resultando no uso de áudio e vídeo por demanda em um bilhão de máquinas. Sob essas circunstâncias, ficou claro que o IP precisava evoluir para se tornar mais flexível.

Em 1990, com esses problemas no horizonte, a IETF começou a trabalhar em uma nova versão do IP, capaz de impedir que os endereços fossem esgotados e de resolver uma série de outros problemas, além de ser mais flexível e mais eficiente. Aqui estão seus principais objetivos:

1. Aceitar bilhões de hosts, mesmo com alocação de espaço de endereços ineficiente.
2. Reduzir o tamanho das tabelas de roteamento.
3. Simplificar o protocolo, de modo a permitir que os roteadores processem os pacotes com mais rapidez.

4. Oferecer mais segurança (autenticação e privacidade) do que o IP atual.
5. Dar mais importância ao tipo de serviço, particularmente no caso de dados em tempo real.
6. Permitir multidifusão, possibilitando a especificação de escopos.
7. Permitir que um host mude de lugar sem precisar mudar o endereço.
8. Permitir que o protocolo evolua no futuro.
9. Permitir a coexistência entre protocolos novos e antigos durante anos.

Para chegar a um protocolo que atendesse a todos esses requisitos, a IETF convocou os interessados a apresentarem suas propostas na RFC 1550. Foram recebidas 21 respostas, mas nem todas foram consideradas propostas completas. Em dezembro de 1992, havia sete propostas muito interessantes em estudo. As propostas variavam desde pequenos ajustes no IP à sua eliminação pura e simples, com a criação de um protocolo totalmente diferente.

Uma proposta era executar o TCP sobre o CLNP que, com seus endereços de 160 bits, seria capaz de oferecer um espaço de endereços infinito e unificaria os dois principais protocolos da camada de rede. No entanto, para muita gente, isso seria o mesmo que admitir que o mundo OSI ainda tinha suas vantagens, uma afirmação politicamente incorreta nos círculos da Internet. A padronização do protocolo CLNP tinha características muito parecidas com a do IP; portanto, não podemos afirmar que os dois protocolos sejam de fato muito diferentes. Na verdade, o protocolo que acabou sendo escolhido apresenta muito mais diferenças em relação ao IP do que o CLNP. Um dos fatores que pesaram contra o CLNP foi a baixa qualidade em relação aos tipos de serviços oferecidos, algo de fundamental importância para uma eficiente transmissão multimídia.

As três melhores propostas foram publicadas na *IEEE Network* (Deering, 1993; Francis, 1993; e Katz e Ford, 1993). Depois de muita discussão, revisão e disputa, foi selecionada uma versão combinada modificada das propostas de Deering e

Francis, agora chamada **SIPP (Simple Internet Protocol Plus)**, à qual foi atribuída a designação **IPv6.**

O IPv6 atende a todos os objetivos propostos, preservando os bons recursos do IP, descartando ou reduzindo a importância das características ruins e criando outras quando necessário. Genericamente, o IPv6 não é compatível com o IPv4, mas é compatível com todos os outros protocolos auxiliares da Internet, incluindo TCP, UDP, ICMP, IGMP, OSPF, BGP e DNS, apesar de, em certos momentos, serem necessárias pequenas modificações (principalmente quando têm de lidar com endereços mais longos). Os principais recursos do IPv6 serão discutidos a seguir. Para obter mais informações sobre ele, consulte as RFCs 2460 a 2466.

Em primeiro lugar, o IPv6 tem endereços mais longos que o IPv4. Eles têm 16 bytes, o que resolve o problema que o IPv6 se propõe resolver: oferecer um número ilimitado de endereços na Internet. Voltaremos a descrever os endereços mais adiante.

O segundo aperfeiçoamento importante no IPv6 é a simplificação do cabeçalho. Ele contém apenas 7 campos (contra os 13 do IPv4). Essa mudança permite aos roteadores processarem os pacotes com mais rapidez e, dessa forma, melhorar o throughput e o retardo. Também voltaremos a descrever o cabeçalho em breve.

A terceira mudança importante foi o melhor suporte para as opções oferecidas. Essa mudança era fundamental para o novo cabeçalho, pois os campos que até então eram obrigatórios agora são opcionais. Além disso, é diferente a forma como as opções são representadas, o que torna mais simples para os roteadores ignorar as opções a que eles não se propõem. Esse recurso diminui o tempo de processamento de pacotes.

Uma quarta área em que o IPv6 representa um grande avanço é a segurança. A IETF já estava farta de ver reportagens nos jornais com meninos precoces de 12

anos que, utilizando seus computadores pessoais, conseguiam devassar segredos de grandes instituições financeiras e militares pela Internet. Havia uma forte sensação de que era preciso fazer algo para melhorar a segurança. A autenticação e a privacidade são recursos importantes do novo IP. Porém, essas características foram integradas mais tarde ao IPv4; assim, na área de segurança não há mais diferenças tão grandes.

Por fim, foi dada maior atenção à qualidade de serviço. Diversos esforços corajosos foram feitos no passado; porém, com o crescimento atual da multimídia na Internet, a sensação de urgência é maior.

[T4] O cabeçalho principal do IPv6

O cabeçalho do IPv6 é mostrado na Figura 5.68. O campo *Version* é sempre 6 para o IPv6 (e 4 para o IPv4). Durante o período de transição do IPv4, que provavelmente durará uma década, os roteadores serão capazes de examinar esse campo para identificar o tipo de pacote que eles têm. A propósito, a realização desse teste desperdiça algumas instruções no caminho crítico; portanto, muitas implementações provavelmente tentarão evitá-lo usando algum campo no cabeçalho de enlace de dados para diferenciar os pacotes IPv4 dos pacotes IPv6. Dessa forma, os pacotes podem ser passados diretamente para o tratador da camada de rede correto. No entanto, a estratégia de fazer com que a camada de enlace de dados tenha conhecimento dos tipos de pacotes de rede viola completamente o princípio de projeto, segundo o qual cada camadas não deve saber o significado dos bits que estão sendo passados a ela pela camada superior. As discussões entre os que desejam "fazer o que é certo" e os que querem "tornar o processo mais rápido" ainda deverá se estender por um longo tempo e será motivo de muita polêmica.

[arte: ver original p. 467]

[Dísticos]

[1]32 Bits

[2]Version Traffic class Flow label

[3]Payload length Next header Hop limit

[4]Source address (16 bytes)

[5]Destination address (16 bytes)

[F]Figura 5.68

[FL] O cabeçalho fixo do IPv6 (obrigatório)

O campo *Traffic class* é usado para fazer distinção entre pacotes com diferentes requisitos de entrega em tempo real. Havia um campo destinado a esse propósito no IP desde o início, mas ele só foi implementado esporadicamente por roteadores. Agora estão sendo realizadas experiências para definir a melhor maneira de usá-lo para transmissão de multimídia.

O campo *Flow label* ainda está em fase de experiência, mas será usado para permitir que uma origem e um destino configurem uma pseudoconexão com propriedades e necessidades específicas. Por exemplo, um fluxo de pacotes entre um processo de um determinado host de origem e um certo processo de um host de destino específico pode ter severas restrições em termos de retardo e, por essa razão, necessitar de uma largura de banda reservada. O fluxo pode ser configurado com antecedência e ter um identificador atribuído a ele. Quando um pacote com o campo *Flow label* com valor diferente de zero aparece, todos os roteadores podem verificar nas tabelas internas que tipo de tratamento especial ele exige. Na prática, os fluxos são uma tentativa de se ter a flexibilidade de uma sub-rede de datagramas juntamente com as garantias de uma sub-rede de circuitos virtuais.

Cada fluxo é designado pelo endereço de origem, endereço de destino e número

de fluxo. Portanto, pode haver muitos fluxos ativos ao mesmo tempo entre um determinado par de endereços IP. Por essa razão, quando dois fluxos enviados por diferentes hosts e com o mesmo número de fluxo passarem pelo mesmo roteador, este será capaz de distingui-los usando os endereços de origem e de destino. Para que os roteadores possam analisar os números de fluxo com mais facilidade, eles serão escolhidos ao acaso, em vez de serem atribuídos seqüencialmente a partir de 1.

O campo *Payload length* determina o número de bytes que seguem o cabeçalho de 40 bytes da Figura 5.68. O nome desse campo, que no IPv4 era *Total length*, foi alterado devido à pequena mudança de significado a que foi submetido: os 40 bytes do cabeçalho deixaram de ser contados como parte do tamanho, como acontecia até então.

O campo *Next header* revela um segredo. O cabeçalho pode ser simplificado, porque existe a possibilidade de haver outros cabeçalhos de extensão (opcionais). Esse campo informa quais dos seis cabeçalhos de extensão (atuais) seguem esse cabeçalho, se houver algum. Se esse cabeçalho for o último cabeçalho do IP, o campo *Next header* revelará para qual tratador de protocolo de transporte (por exemplo, TCP, UDP) o pacote deverá ser enviado.

O campo *Hop limit* é usado para impedir que os pacotes tenham duração eterna. Na prática, ele é igual ao campo *Time to live* do IPv4, ou seja, um campo que é decrementado a cada hop. Teoricamente, no IPv4 ele denotava um tempo em segundos, mas nenhum roteador o utilizava dessa maneira. Por esse motivo, seu nome foi alterado para refletir o modo como de fato ele é usado.

Em seguida, vêm os campos *Source address* e *Destination address*. A proposta original de Deering, o SIP, utilizava endereços de 8 bytes; porém, durante o processo de revisão, muitas pessoas perceberam que, com endereços de 8 bytes, o IPv6 esgotaria os endereços disponíveis em apenas algumas décadas, enquanto

os endereços de 16 bytes nunca se esgotariam. Outras pessoas afirmavam que 16 bytes seriam um exagero, e outras alegavam que endereços de 20 bytes seriam compatíveis com o protocolo de datagramas do OSI. Ainda outra facção queria endereços de tamanho variável. Depois de muita discussão, chegou-se à conclusão de que a melhor opção era utilizar endereços de 16 bytes.

Foi criada uma nova notação para representar endereços de 16 bytes. Eles são escritos sob a forma de oito grupos de quatro dígitos hexadecimais, separados por sinais de dois-pontos entre os grupos, como no exemplo a seguir:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Tendo em vista que vários endereços conterão muitos zeros, foram autorizadas três otimizações. Em primeiro lugar, os zeros à esquerda dentro de um grupo podem ser omitidos, de modo que 0123 possa ser escrito como 123. Em segundo lugar, um ou mais grupos de 16 bits zero podem ser substituídos por um par de sinais de dois-pontos. Conseqüentemente, o endereço anterior pode ser escrito da seguinte maneira:

8000::123:4567:89AB:CDEF

Por fim, os endereços IPv4 podem ser escritos empregando-se um par de sinais de dois-pontos e um número decimal tradicional, como neste exemplo:

::192.31.20.46

Talvez não seja necessário ser tão explícito em relação a isso, mas existem muitos endereços de 16 bytes. Especificamente, existem 2^{128} endereços desse tipo, o que significa cerca de 3×10^{38}. Se colocássemos um computador em cada pedaço de terra e água do nosso planeta, o IPv6 permitiria 7×10^{23} endereços IP por metro quadrado. Os estudantes de química perceberão que esse número é maior que o número de Avogadro. Embora não exista a intenção de dar a cada molécula na superfície da Terra seu próprio endereço IP, não estamos longe de chegar a essa marca.

Na prática, o espaço de endereços não será usado com eficiência, exatamente como acontece com o espaço de endereços dos números telefônicos (o código de área de Manhattan, 212, está próximo da saturação, mas o de Wyoming, 307, está quase vazio). Na RFC 3194, Durand e Huitema calcularam que, usando-se a alocação dos números de telefones como um guia, mesmo considerando-se a hipótese mais pessimista, ainda assim haverá mais de 1000 endereços IP por metro quadrado de toda a superfície da Terra (incluindo rios e mares). Em qualquer situação provável, haverá trilhões deles por metro quadrado. Em resumo, parece improvável que eles venham a se esgotar em um futuro próximo.

É interessante comparar o cabeçalho do IPv4 (Figura 5.53) com o cabeçalho do IPv6 (Figura 5.68) e ver o que foi mantido e o que foi descartado no IPv6. O campo *IHL* foi eliminado, porque o cabeçalho do IPv6 tem um tamanho fixo. O campo *Protocol* foi retirado porque o campo *Next header* identifica o que vem depois do último cabeçalho IP (por exemplo, um segmento UDP ou TCP).

Todos os campos relacionados à fragmentação foram removidos, porque o IPv6 dá um tratamento diferente à fragmentação. Para começar, todos os hosts e roteadores compatíveis com o IPv6 devem determinar dinamicamente o tamanho de datagrama que será usado. Essa regra diminui a probabilidade de ocorrer fragmentação. O valor mínimo também foi elevado de 576 para 1280, a fim de permitir o uso de 1024 bytes de dados e muitos cabeçalhos. Além disso, quando um host enviar um pacote IPv6 muito grande, o roteador que não puder encaminhá-lo, enviará de volta uma mensagem de erro em vez de fragmentá-lo. Essa mensagem instrui o host a dividir todos os novos pacotes enviados a esse destino. Na verdade, é muito mais eficiente obrigar o host a enviar pacotes com o tamanho exato que solicitar que os roteadores os fragmentem automaticamente.

Por fim, o campo *Checksum* foi eliminado, porque esse cálculo reduz de forma significativa o desempenho. Com as redes confiáveis usadas atualmente, além do

fato de a camada de enlace de dados e as camadas de transporte terem seus próprios totais de verificação, a importância de um novo total é insignificante, se comparada com a queda de desempenho que ela implica. Com a remoção de todos esses recursos, o protocolo da camada de rede ficou muito mais enxuto e prático. Assim, o objetivo do IPv6 — um protocolo a um só tempo rápido e flexível, capaz de oferecer um grande espaço de endereços — foi atendido por esse projeto.

[T4] Cabeçalhos de extensão

Ocasionalmente, alguns dos campos ausentes do IPv4 ainda serão necessários; assim, o IPv6 introduziu o conceito de um **cabeçalho de extensão** (opcional). Esses cabeçalhos podem ser criados com a finalidade de oferecer informações extras, desde que elas sejam codificadas de maneira eficiente. Atualmente, há seis tipos de cabeçalhos de extensão definidos, mostrados na Figura 5.69. Todos eles são opcionais mas, se houver mais de um, eles terão de aparecer logo depois do cabeçalho fixo e, de preferência, na ordem listada.

[arte: ver original p. 470a]

[T]Tabela

Cabeçalho de extensão	**Descrição**
Hop-by-hop options	Informações diversas para os roteadores
Destination options	Informações adicionais para o destino
Routing	Lista parcial de roteadores a visitar
Fragmentation	Gerenciamento de fragmentos de datagramas
Authentication	Verificação da identidade do transmissor
Encrypted security payload	Informações sobre o conteúdo criptografado

[F]Figura 5.69

[FL] Cabeçalhos de extensão do IPv6

Alguns desses cabeçalhos têm um formato fixo; outros contêm um número variável de campos de comprimento variável. Nesses casos, cada item é codificado como uma tupla (Type, Lenght, Value). *Type* é um campo de 1 byte que identifica a opção. Os valores de *Type* foram escolhidos de tal forma que os dois primeiros bits informem o que devem fazer os roteadores que não sabem como processar a opção. Estas são as possibilidades: ignorar a opção, descartar o pacote, descartar o pacote e enviar de volta um pacote ICMP, e ainda a mesma opção anterior sem que, no entanto, sejam enviados pacotes ICMP para endereços de multidifusão (para impedir que um pacote de multidifusão defeituoso gere milhões de relatórios ICMP).

Length também é um campo de 1 byte. Ele identifica o tamanho do valor (0 a 255 bytes). *Value* contém todas as informações obrigatórias, com no máximo 255 bytes.

O cabeçalho hop-by-hop é usado para as informações que todos os roteadores ao longo do caminho devem examinar. Até agora, uma opção foi definida: compatibilidade com datagramas além de 64 K. O formato desse cabeçalho é mostrado na Figura 5.70. Quando ele é usado, o campo *Payload length* do cabeçalho fixo é definido como zero.

[arte: ver original p. 470b]

[Dísticos]

[1]Next header 0 194 0

[2]Jumbo payload length

[F]Figura 5.70

[FL] O cabeçalho de extensão hop-by-hop para datagramas grandes (jumbogramas)

A exemplo de todos os outros cabeçalhos de extensão, esse cabeçalho começa com um byte, cuja função é identificar o tipo de cabeçalho que vem a seguir. Depois desse byte, há um byte cuja função é identificar o tamanho do cabeçalho hop-by-hop, excluindo os primeiros 8 bytes, que são obrigatórios. Todas as extensões começam dessa maneira.

Os 2 bytes seguintes indicam que essa opção define o tamanho do datagrama (código 194) e que o tamanho é um número de 4 bytes. Os 4 últimos bytes mostram o tamanho do datagrama. Não são permitidos datagramas com menos de 65.536 bytes; datagramas maiores resultarão na eliminação do pacote no primeiro roteador, e no envio de uma mensagem de erro do ICMP. Os datagramas que utilizam essa extensão de cabeçalho são chamados **jumbogramas.** O uso de jumbogramas é importante para as aplicações de supercomputadores, que devem transferir gigabytes de dados pela Internet com grande eficiência.

O cabeçalho de opções de destino é usado em campos que só precisam ser interpretados no host de destino. Na versão inicial do IPv6, as únicas opções definidas são opções nulas para preencher esse cabeçalho até formar um múltiplo de oito bytes; portanto, ele não será utilizado inicialmente. Esse campo foi incluído para garantir que o novo software de roteamento e de host poderá tratá-lo, no caso de alguém imaginar uma opção de destino algum dia.

O cabeçalho de roteamento lista um ou mais roteadores que devem ser visitados no caminho até o destino. Ele é muito semelhante ao roteamento de origem livre do IPv4, no fato de todos os endereços listados terem de ser visitados em ordem; porém, outros roteadores não listados também podem ser visitados. O formato do cabeçalho de roteamento é mostrado na Figura 5.71.

[arte: ver original p. 471]

[Dísticos]

[1]Next header Header extension length Routing type Segments

left

[2]Dados específicos do tipo

[F]Figura 5.71

[FL] O cabeçalho de extensão para roteamento

Os 4 primeiros bytes do cabeçalho de extensão para roteamento contêm quatro inteiros de 1 byte: os campos *Next header* e *Header extension length* já foram descritos. O campo *Routing type* fornece o formato do restante do cabeçalho. O tipo 0 informa que uma palavra reservada de 32 bits segue a primeira palavra, e é acompanhada por algum número de endereços IPv6. Outros tipos podem ser criados no futuro, se necessário. Por fim, o campo *Segments left* controla quantos endereços da lista ainda não foram visitados. Ele é decrementado toda vez que um endereço é visitado. Quando chega a 0, o pacote fica por sua própria conta, sem nenhuma orientação adicional sobre qual rota seguir. em geral, a essa altura ele está tão perto do destino que a melhor rota é evidente.

O cabeçalho de fragmento lida com a fragmentação da mesma maneira que o IPv4. O cabeçalho contém o identificador do datagrama, o número do fragmento e um bit que informa se haverá novos fragmentos em seguida. No IPv6, ao contrário do IPv4, apenas o host de origem pode fragmentar um pacote. Os roteadores ao longo do caminho não podem fazê-lo. Embora represente uma grande ruptura filosófica com o passado, esse recurso simplifica o trabalho dos roteadores e faz com que o roteamento seja mais rápido. Como já dissemos, se um roteador for confrontado com um pacote muito grande, ele o descartará e enviará um pacote ICMP de volta à origem. Tais informações permitem que o host de origem utilize esse cabeçalho para fragmentar o pacote em pedaços menores e tentar outra vez.

O cabeçalho de autenticação oferece um mecanismo pelo qual o receptor de um

pacote pode ter certeza de quem o enviou. A carga útil de segurança criptografada torna possível criptografar o conteúdo de um pacote, de modo que apenas o destinatário pretendido possa ler seu conteúdo. Esses cabeçalhos utilizam técnicas criptográficas para alcançar os objetivos a que se propõem.

[T4] Controvérsias

Considerando-se o processo de projeto aberto e a paixão com que as pessoas defenderam suas opiniões, não foi à toa que muitas das escolhas feitas para o IPv6 tenham gerado tanta polêmica. Vamos apresentar um breve resumo dessas controvérsias. Se desejar conhecer todos os detalhes, consulte as RFCs relevantes.

Já mencionamos o argumento sobre o tamanho do endereço. O resultado, fruto de uma conciliação, foi o uso de endereços de comprimento fixo de 16 bytes.

O tamanho do campo *Hop limit* também gerou muita discussão. De um lado, estavam os defensores da tese de que seria um grande equívoco limitar o número de hops a um máximo de 255 (implícito na utilização de um campo de 8 bits). Afinal de contas, os caminhos de 32 hops são comuns agora e, daqui a 10 anos, talvez sejam comuns caminhos muito mais longos. As pessoas argumentaram que a utilização de um tamanho de endereço gigantesco era uma boa idéia, mas usar um pequeno número de hops limitava as opções. Para elas, o maior pecado que a ciência da computação poderia cometer seria oferecer tão poucos bits.

Do outro lado, estavam os que acreditavam que a ampliação excessiva do campo incharia o cabeçalho. Além disso, a função do campo *Hop limit* é impedir que os pacotes vagueiem por muito tempo, tese incompatível com o tempo necessário para percorrer 65.535 hops. Por fim, com o crescimento da Internet, serão criados cada vez mais enlaces de longa distância, tornando possível ir de qualquer país a qualquer outro país, usando no máximo meia dezena de hops. Se

forem necessários mais de 125 hops para ir da origem até o destino através de seus respectivos gateways internacionais, algo estará errado com os backbones nacionais. Os defensores dos 8 bits venceram essa batalha.

Outro problema complicado era o tamanho máximo do pacote. A comunidade dos supercomputadores desejava pacotes com mais de 64 KB. Quando inicia a transferência, um supercomputador está tentando executar uma tarefa importantíssima e não deve ser interrompido a cada 64 KB. O argumento contrário ao uso de grandes pacotes é: se um pacote de 1 MB alcançar uma linha T1 de 1,5 Mbps, ele a ocupará durante 5 segundos, o que produzirá um retardo bastante perceptível para os usuários interativos que estão compartilhando a linha. Chegou-se a um acordo quanto a essa questão: os pacotes normais foram limitados a 64 KB, mas o cabeçalho de extensão hop-by-hop pode ser usado para permitir a utilização de jumbogramas.

Outro assunto polêmico foi a remoção do total de verificação do IPv4. Algumas pessoas comparavam esse movimento à remoção dos freios de um automóvel. Dessa forma, o veículo ficaria mais leve e mais veloz mas, se ocorresse alguma situação inesperada, haveria sérios problemas.

O argumento contrário aos totais de verificação levava em conta que, qualquer aplicação que realmente se preocupasse com a integridade dos dados teria de incluir um total de verificação na camada de transporte e, por essa razão, seria uma redundância ter um novo total de verificação no IP (além do total de verificação da camada de enlace de dados). Vale lembrar também que a experiência mostrava que o cálculo do total de verificação do IP representava um grande desperdício no IPv4. Os que eram contrários ao total de verificação venceram essa batalha, e o IPv6 ficou sem o total de verificação.

Os hosts móveis também foram motivo de discórdia. Quando um computador portátil vai de um canto a outro do mundo, ele pode continuar a operar no

destino com o mesmo endereço IPv6, ou tem de usar um esquema com agentes nacionais e internacionais? Os hosts móveis também introduzem assimetrias no sistema de roteamento. É bastante provável que um pequeno computador móvel possa captar facilmente o potente sinal transmitido por um grande roteador estacionário, mas talvez o roteador estacionário não seja capaz de captar o tênue sinal transmitido pelo host móvel. Conseqüentemente, algumas pessoas quiseram dar ao IPv6 uma compatibilidade explícita com hosts móveis. Esse esforço fracassou, pois não foi possível chegar a um consenso quanto a essa questão.
Provavelmente, a maior batalha se deu no campo da segurança. Todos concordavam que ela era essencial. O problema descobrir onde e como usar a segurança. Primeiro, onde. O argumento para inseri-la na camada de rede é que nessa camada ela se torna um serviço padrão que todas as aplicações podem usar sem qualquer planejamento prévio. O argumento contrário é que, em geral, tudo o que as aplicações realmente seguras desejam é a criptografia fim a fim, na qual a aplicação de origem cuida da codificação, e a aplicação de destino a desfaz. Se não for assim, o usuário estará à mercê de implementações potencialmente problemáticas, sobre as quais ele não tem o menor controle, na camada de rede.
A resposta a esse argumento é que essas aplicações podem se abster de usar os recursos de segurança do IP, executando elas mesmas essa tarefa. A réplica a esse argumento é que as pessoas que não confiam na execução adequada dessa tarefa e não estão dispostas a pagar o preço das desajeitadas e lentas implementações IP que dispõem desse recurso, ainda que ele esteja desativado.
Outro aspecto a ser levado em consideração quanto à localização da segurança diz respeito ao fato de que, em muitos países (mas não em todos) as leis de exportação envolvendo a criptografia são muito rígidas. Alguns deles, como a França e o Iraque, também impõem inúmeras restrições quanto a seu uso doméstico; portanto, as pessoas não podem guardar segredos.

Conseqüentemente, os Estados Unidos (e muitos outros países) não teriam mercado consumidor para qualquer implementação do IP que usasse um sistema criptográfico suficientemente forte para ser digno de valor. A maioria dos fabricantes de computadores detesta produzir dois conjuntos de software, um para uso doméstico e outro para exportação.

Um ponto sobre o qual não houve controvérsia foi o fato de ninguém esperar que a Internet do IPv4 passasse da noite para o dia a ser a Internet do IPv6. Em vez disso, "ilhas" de IPv6 isoladas serão convertidas, comunicando-se inicialmente através de túneis. Quando começarem a se desenvolver, as ilhas do IPv6 formarão ilhas maiores. Em algum momento, todas as ilhas estarão unidas, e a Internet estará totalmente convertida. Devido ao maciço investimento feito há pouco tempo em roteadores IPv4, o processo de conversão deve durar pelo menos uma década. Por essa razão, houve um grande esforço no sentido de garantir que essa transição ocorra da maneira menos dolorosa possível. Para obter mais informações sobre o IPv6, consulte (Loshin, 1999).

[T2] 5.7 Resumo

A camada de rede fornece serviços à camada de transporte. Ela pode se basear em circuitos virtuais ou datagramas. Em ambos os casos, sua principal tarefa é rotear pacotes da origem até o destino. Nas sub-redes de circuitos virtuais, uma decisão de roteamento é tomada quando o circuito virtual é configurado. Nas sub-redes de datagramas, essa decisão é tomada a cada pacotes.

Muitos algoritmos de roteamento são usados nas redes de computadores. Os algoritmos estáticos incluem o roteamento pelo caminho mais curto e a inundação. Os algoritmos dinâmicos incluem o roteamento com vetor de distância e o roteamento de estado de enlace. A maioria das redes reais utiliza um desses algoritmos. Outros assuntos importantes relacionados ao roteamento são o

roteamento hierárquico, o roteamento de hosts móveis, o roteamento por difusão, o roteamento por multidifusão e o roteamento em redes não hierárquicas.

As sub-redes podem se tornar congestionadas, aumentando o retardo e reduzindo o throughput dos pacotes. Os projetistas de rede tentam evitar o congestionamento através de um projeto adequado. As técnicas incluem a política de retransmissão, o armazenamento em caches, o controle de fluxo e outras. Se ocorrer um congestionamento, ele terá de ser administrado. Os pacotes reguladores podem ser enviados de volta, a carga pode ser escoada e outros métodos podem ser aplicados.

A próxima etapa além de lidar com o congestionamento é tentar de fato alcançar uma qualidade de serviço garantida. Os métodos que podem ser usados para isso incluem o armazenamento em buffers no cliente, a moldagem do tráfego, a reserva de recursos e o controle de admissão. Entre as abordagens adotadas para se obter boa qualidade de serviço estão os serviços integrados (incluindo o RSVP), os serviços diferenciados e o MPLS.

As redes apresentam diferenças em vários aspectos; portanto, podem ocorrer problemas quando várias redes estão conectadas. Às vezes, os problemas podem ser superados efetuando-se o tunneling quando um pacote passa por uma rede hostil mas, se as redes de origem e de destino forem diferentes, essa estratégia não funcionará. Quando diferentes redes tiverem diferentes tamanhos máximos de pacotes, será possível recorrer à sua fragmentação.

A Internet tem uma rica variedade de protocolos relacionados à camada de rede. Entre eles, encontram-se o protocolo de transporte de dados, o IP, os protocolos de controle ICMP, ARP e RARP, e os protocolos de roteamento OSPF e BGP. A Internet está esgotando rapidamente os endereços IP, e foi desenvolvida uma nova versão do IP, o IPv6, para resolver esse problema.

[T2] Problemas

1. Cite dois exemplos de aplicações para as quais é apropriado um serviço orientado a conexões. Depois, cite dois exemplos para os quais é mais apropriado um serviço sem conexões.

2. Há alguma circunstância em que o serviço orientado a conexões entregará (ou, pelo menos, deveria entregar) os pacotes fora de ordem? Explique.

3. As sub-redes de datagramas roteiam cada pacote como uma unidade separada, independente das demais. As sub-redes de circuitos virtuais não precisam tomar essa providência, pois os pacotes de dados seguem uma rota predeterminada. Você diria que essa observação significa que as sub-redes de circuitos virtuais não precisam da capacidade de rotear pacotes isolados de uma origem arbitrária até um destino arbitrário? Explique sua resposta.

4. Cite três exemplos de parâmetros de protocolos que devem ser negociados quando uma conexão é configurada.

5. Considere o seguinte problema de projeto relacionado à implementação do serviço de circuitos virtuais: se os circuitos virtuais forem usados dentro da sub-rede, cada pacote de dados deverá ter um cabeçalho de 3 bytes, e cada roteador deverá vincular até 8 bytes de armazenamento para identificação do circuito. Se forem usados internamente, os datagramas precisarão de cabeçalhos de 15 bytes, mas não haverá necessidade de qualquer espaço de tabela de roteador. A capacidade de transmissão custa 1 centavo por 10^6 bytes em cada hop. A memória do roteador pode ser comprada a 1 centavo por byte e é depreciada em dois anos, considerando-se apenas o horário comercial, com uma semana de 40 horas. Estatisticamente, a sessão média é executada durante 1000 segundos, e nesse tempo são transmitidos 200 pacotes. O pacote médio exige quatro hops. Qual é a implementação mais econômica e quanto ela custa?

6. Como todos os roteadores e hosts estão funcionando adequadamente e todo o software está isento de todos os erros, há alguma chance, por menor que seja, de que um pacote seja entregue ao destino errado?

7. Considere a rede da Figura 5.7, mas ignore os pesos nas linhas. Suponha que ela utilize a inundação como algoritmo de roteamento. Se um pacote enviado por *A* até *D* tem uma contagem máxima de hops igual a 3, liste todas as rotas que ele seguirá. Além disso, informe quantos hops o pacote consome de largura de banda.

8. Cite uma heurística simples para localizar dois caminhos através de uma rede de uma determinada origem para um determinado destino que possa sobreviver à perda de qualquer linha de comunicação (desde que existam dois desses caminhos). Os roteadores são considerados suficientemente confiáveis; portanto, não é preciso se preocupar com a possibilidade de panes em roteadores.

9. Considere a sub-rede da Figura 5.13(a). O roteamento com vetor de distância é usado e os vetores a seguir acabaram de entrar no roteador *C*: De *B*: (5, 0, 8, 12, 6, 2), de *D*: (16, 12, 6, 0, 9, 10); e de *E*: (7, 6, 3, 9, 0, 4). Os retardos medidos para *B*, *D* e *E* são 6, 3 e 5, respectivamente. Qual é a nova tabela de roteamento de *C*? Forneça a linha de saída a ser usada e o retardo esperado.

10. Se os retardos forem registrados como números de 8 bits em uma rede com 50 roteadores e os vetores de retardo forem trocados duas vezes por segundo, qual será a largura de banda por linha (full-duplex) ocupada pelo algoritmo de roteamento distribuído? Parta do princípio de que cada roteador tem três linhas para outros roteadores.

11. Na Figura 5.14, o OR booleano dos dois conjuntos de bits ACF é 111 em cada linha. Trata-se de uma coincidência ou ele é mantido em todas as sub-redes, independente das circunstâncias?

12. No roteamento hierárquico com 4800 roteadores, que tamanho de região e de

agrupamento deve ser escolhido para minimizar o tamanho da tabela de roteamento no caso de uma hierarquia de três camadas? Um bom ponto de partida é a hipótese de que uma solução com *k* agrupamentos de *k* regiões com *k* roteadores está próxima de ser ótima; isso significa que *k* é aproximadamente a raiz cúbica de 4800 (cerca de 16). Utilize um processo de tentativa e erro para verificar combinações em que todos os três parâmetros estão na vizinhança genérica de 16.

13. No texto foi declarado que, quando um host móvel não está em sua localização de origem, os pacotes enviados para sua LAN são interceptados pelo agente local dessa LAN. Em uma rede IP de uma LAN 802.3, de que maneira o agente local executa essa interceptação?

14. Observando a sub-rede da Figura 5.6, quantos pacotes são gerados por uma difusão de *B*, usando-se:

(a) Encaminhamento pelo caminho inverso?

(b) A árvore de escoamento?

15. Considere a rede da Figura 5.16(a). Imagine que uma nova linha seja acrescentada entre *F* e *G*, mas que a árvore de escoamento da Figura 5.16(b) permaneça inalterada. Que mudanças ocorrerão na Figura 5.16(c)?

16. Calcule a árvore de amplitude de multidifusão para o roteador *C* na sub-rede a seguir para um grupo com membros nos roteadores *A*, *B*, *C*, *D*, *E*, *F*, *I* e *K*.

[arte: ver original p. 476]

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

17. Na Figura 5.20, verifique se o nós *H* ou *I* transmite por difusão na pesquisa que se inicia em *A*.

18. Suponha que o nó *B* na Figura 5.20 tenha acabado de ser reinicializado e não tenha nenhuma informação de roteamento em suas tabelas. De repente, ele precisa de uma rota para *H*. O nó transmite difusões com *TTL* definido como 1, 2, 3 e assim por diante. Quantas rodadas ele levará para encontrar uma rota?
19. Na versão mais simples do algoritmo Chord para pesquisa não hierárquica, as pesquisas não utilizam a tabela finger. Em vez disso, elas são lineares em torno do círculo, em qualquer sentido. Um nó poderia prever com exatidão em que sentido deve pesquisar? Explique.
20. Considere o círculo de Chord da Figura 5.24. Suponha que o nó 10 se conecte repentinamente. Isso afetará a tabela finger do nó 1? Em caso afirmativo, como?
21. Como um possível mecanismo de controle de congestionamento em uma sub-rede que utiliza circuitos virtuais internamente, um roteador poderia privar-se de confirmar um pacote recebido até (1) saber que sua última transmissão ao longo do circuito virtual foi recebida com sucesso e (2) ter um buffer livre. Para simplificar, parta do princípio de que os roteadores usam um protocolo stop-and-wait e que cada circuito virtual tem um buffer dedicado a ele em cada sentido do tráfego. Se ele precisar de *T* segundos para transmitir um pacote (dados ou confirmação) e houver *n* roteadores no caminho, qual será a taxa em que os pacotes serão entregues ao host de destino? Suponha que os erros de transmissão sejam raros e que a conexão entre host e roteador seja infinitamente rápida.
22. Uma sub-rede de datagramas permite que os roteadores eliminem pacotes sempre que precisarem. A probabilidade de um roteador descartar um pacote é *p*. Considere o caso de um host de origem conectado ao roteador de origem, que está conectado ao roteador de destino que, por sua vez, está conectado ao host de destino. Se um dos roteadores descartar um pacote, o host de origem sofrerá um timeout e fará novas tentativas. Se as linhas host-roteador e roteador-

roteador fossem contadas como hops, qual seria o número médio de:

(a) Hops que um pacote executa por transmissão?

(b) Transmissões que um pacote cria?

(c) Hops necessários por pacote recebido?

23. Descreva duas diferenças importantes entre o método de bit de advertência e o método RED.

24. Cite um argumento para que o algoritmo de balde furado permita apenas um pacote por pulso, independente do tamanho do pacote.

25. A variante de contagem de bytes do algoritmo de balde furado é usada em um determinado sistema. A regra é que um pacote de 1024 bytes, dois pacotes de 512 bytes etc. possam ser enviados em cada pulso. Cite uma séria restrição desse sistema que não foi mencionada no texto.

26. Uma rede ATM utiliza um esquema de balde de símbolos para moldagem de tráfego. Um novo símbolo é colocado no balde a cada 5 μs. Cada símbolo se refere a uma célula, que contém 48 bytes de dados. Qual é a taxa de dados máxima sustentável?

27. Um computador de uma rede de 6 Mbps é controlado por um balde de símbolos. O balde de símbolos é preenchido a uma taxa de 1 Mbps. Inicialmente, sua capacidade é de 8 megabits. Durante quanto tempo o computador pode transmitir a 6 Mbps?

28. Imagine uma especificação de fluxo que tem um tamanho máximo de pacote de 1000 bytes, uma taxa de balde de símbolos de 10 milhões de bytes/s, um tamanho de balde de símbolos de 1 milhão de bytes e uma taxa máxima de transmissão de 50 milhões de bytes/s. Quanto tempo poderá durar uma rajada na velocidade máxima?

29. A rede da Figura 5.37 utiliza o RSVP com árvores de multidifusão nos hosts 1 e 2 como mostra a figura. Suponha que o host 3 solicite um canal de largura de

banda de 2 MB/s para um fluxo proveniente do host 1 e outro canal de largura de banda de 1 MB/s para um fluxo do host 2. Ao mesmo tempo, o host 4 solicita um canal de largura de banda igual a 2 MB/s para um fluxo do host 1, e o host 5 solicita um canal de largura de banda de 1 MB/s para um fluxo vindo do host 2. Qual será a largura de banda total reservada para essas solicitações nos roteadores *A*, *B*, *C*, *E*, *H*, *J*, *K* e *L*?

30. A CPU de um roteador pode processar 2 milhões de pacotes/s. A carga oferecida a ela é 1,5 milhões de pacotes/s. Se uma rota da origem até o destino contiver 10 roteadores, quanto tempo será gasto no enfileiramento e no serviço pelas CPUs?

31. Considere o usuário de serviços diferenciados com encaminhamento expedido. Existe alguma garantia de que os pacotes expedidos experimentarão um retardo mais curto que os pacotes regulares? Por que ou por que não?

32. A fragmentação é necessária em inter-redes de circuitos virtuais concatenados, ou apenas em sistemas de datagramas?

33. O tunneling em uma rede com circuitos virtuais concatenados é bastante simples: o roteador multiprotocolo de uma extremidade estabelece um circuito virtual para a outra extremidade e passa os pacotes através dela. Esse recurso também pode ser usado em sub-redes de datagramas? Se puder, de que forma?

34. Suponha que o host *A* esteja conectado a um roteador *R*1, que *R*1 esteja conectado a outro roteador *R*2, e que *R*2 esteja conectado ao host *B*. Suponha que uma mensagem TCP contendo 900 bytes de dados e 20 bytes de cabeçalho TCP seja repassada ao código IP do host *A* para ser entregue a *B*. Mostre os campos *Total length*, *Identification*, *DF*, *MF* e *Fragment offset* do cabeçalho IP em cada pacote transmitido pelos três enlaces. Suponha que o enlace *A*–*R*1 possa admitir um tamanho máximo de quadro de 1024 bytes, incluindo um cabeçalho de quadro de 14 bytes, que o enlace *R*1–*R*2 possa admitir um tamanho máximo de

quadro de 512 bytes, incluindo um cabeçalho de quadro de 8 bytes, e que o enlace *R*2-*B* possa admitir um tamanho máximo de quadro de 512 bytes, incluindo um cabeçalho de quadro de 12 bytes.

35. Um roteador está transmitindo pacotes IP cujo comprimento total (dados mais cabeçalho) é de 1024 bytes. Supondo-se que os pacotes tenham a duração de 10 segundos, qual será a velocidade máxima de linha em que o roteador poderá operar sem o perigo de circular pelo espaço de números de identificação de datagramas do IP?

36. Um datagrama IP usando a opção *Strict source routing* tem de ser fragmentado. Para você, a opção é copiada para cada fragmento ou basta colocá-la no primeiro fragmento? Explique sua resposta.

37. Suponha que, em vez de serem utilizados 16 bits na parte de rede de um endereço da classe B, tenham sido usados 20 bits. Nesse caso, quantas redes da classe B existiram?

38. Converta o endereço IP cuja representação hexadecimal é C22F1582 em uma notação decimal com pontos.

39. A máscara de sub-rede de uma rede na Internet é 255.255.240.0. Qual é o número máximo de hosts que ela pode manipular?

40. Um grande número de endereços IP consecutivos está disponível a partir de 198.16.0.0. Suponha que quatro organizações, *A*, *B*, *C* e *D*, solicitem 4000, 2000, 4000 e 8000 endereços, respectivamente, e nessa ordem. Para cada uma delas, forneça o primeiro endereço IP atribuído, o último endereço IP atribuído e a máscara na notação *w.x.y.z/s*.

41. Um roteador acabou de receber estes novos endereços IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 e 57.6.120.0/21. Se todos usarem a mesma linha de saída, eles poderão ser agregados? Em caso afirmativa, a quê? Em caso negativo, por que não?

42. O conjunto de endereços IP de 29.18.0.0 até 19.18.128.255 foi agregado a 29.18.0.0/17. Porém, existe um intervalo de 1024 endereços não atribuídos, desde 29.18.60.0 até 29.18.63.255 que agora são repentinamente atribuídos a um host que utiliza uma linha de saída diferente. É necessário dividir o endereço agregado em seus blocos constituintes, acrescentar o novo bloco à tabela e depois verificar se é possível uma reagregação? Em caso contrário, o que poderia ser feito?

43. Um roteador tem as seguintes entradas (CIDR) em sua tabela de roteamento:

Endereço/máscara	Próximo hop
135.46.56.0/22	Interface 0
135.46.60.0/22	Interface 1
192.53.40.0/23	Roteador 1
padrão	Roteador 2

Para cada um dos endereços IP a seguir, o que o roteador fará se chegar um pacote com esse endereço?

(a) 135.46.63.10

(b) 135.46.57.14

(c) 135.46.52.2

(d) 192.53.40.7

(e) 192.53.56.7

44. Muitas empresas adotam a política de manter dois (ou mais) roteadores para conectar a empresa à Internet, a fim de proporcionar alguma redundância no caso de um deles ficar inativo. Essa política ainda será possível com a NAT? Explique sua resposta.

45. Você acabou de explicar o que é um protocolo ARP a um amigo. No final, ele diz: "Entendi. Como o ARP fornece um serviço à camada de rede, isso significa que ele faz parte da camada de enlace de dados." O que você diz a ele?

46. O ARP e o RARP mapeiam endereços de um espaço para outro. Nesse sentido, eles são iguais. No entanto, eles apresentam diferenças fundamentais ao serem implementados. Quais são as principais diferenças entre eles?

47. Descreva uma forma de remontar os fragmentos IP no destino.

48. A maioria dos algoritmos de remontagem do datagrama IP tem um timer para evitar que um fragmento perdido seja anexado definitivamente aos buffers de remontagem. Suponha que um datagrama tenha sido dividido em quatro fragmentos. Os três primeiros fragmentos chegam, mas o último deles é retardado. A uma certa altura, o timer é desativado e os três fragmentos contidos na memória do receptor são descartados. Logo depois, chega o último fragmento. O que deve ser feito com ele?

49. No IP e no ATM, o total de verificação abrange apenas o cabeçalho, e não os dados. Para você, por que essa estrutura foi escolhida?

50. Uma pessoa que mora em Boston viaja para Minneapolis levando consigo seu computador portátil. Para sua surpresa, a LAN em Minneapolis é do tipo IP sem fio; portanto, essa pessoa não precisa se conectar. Mesmo assim, será preciso percorrer toda a empresa com agentes locais e agentes externos para fazer com que as mensagens de correio eletrônico e outros tipos de tráfego cheguem corretamente?

51. O IPv6 utiliza endereços de 16 bytes. Se um bloco de 1 milhão de endereços for alocado a cada picossegundo, qual será a duração desses endereços?

52. O campo *Protocol* usado no cabeçalho do IPv4 não é encontrado no cabeçalho fixo do IPv6. Por quê?

53. Quando o protocolo IPv6 é introduzido, o protocolo ARP tem de ser alterado? Se houver necessidade de mudanças, elas serão conceituais ou técnicas?

54. Crie um programa para simular o roteamento usando o algoritmo de inundação. Cada pacote deve conter um contador que é decrementado a cada

hop. Quando o contador chegar a zero, o pacote será descartado. O tempo é discreto, com cada linha tratando de um pacote a cada intervalo de tempo. Crie três versões do programa: todas as linhas são inundadas, todas as linhas com exceção da linha de entrada são inundadas, e somente as k melhores linhas (escolhidas estatisticamente) são inundadas. Compare o algoritmo de inundação com o roteamento determinístico ($k = 1$) em termos de retardo e de largura de banda utilizada.

55. Crie um programa capaz de simular uma rede de computadores usando o tempo discreto. O primeiro pacote de cada fila do roteador cria um hop por intervalo de tempo. Cada roteador tem apenas um número finito de buffers. Se um pacote chegar e não houver espaço, ele será descartado e não será retransmitido. Em vez disso, há um protocolo fim a fim, repleto de intervalos de timeout e pacotes de confirmação, que em algum momento regenera o pacote do roteador de origem. Represente o throughput da rede como uma função do intervalo de timeout fim a fim, parametrizado pela taxa de erros.

56. Crie uma função para realizar o encaminhamento em um roteador IP. O procedimento tem um único parâmetro, um endereço IP. Ele também tem acesso a uma tabela global que consiste em um array de triplas. Cada tripla contém três inteiros: um endereço IP, uma máscara de sub-rede e a linha de saída a ser usada. A função pesquisa o endereço IP na tabela usando o CIDR e retorna como seu valor a linha que deverá ser usada.

57. Use o programa *traceroute* (UNIX) ou o programa *tracert* (Windows) para traçar a rota desde seu computador até várias universidades em outros continentes. Faça uma lista dos enlaces transoceânicos que você descobrir. Alguns sites que você deve tentar visitar são:

www.berkeley.edu (Califórnia)

www.mit.edu (Massachusetts)

www.vu.nl (Amsterdã)

www.ucl.ac.uk (Londres)

www.usyd.edu.au (Sydney)

www.u-tokyo.ac.jp (Tóquio)

www.uct.ac.za (Cidade do Cabo)

[TA1]Capítulo

[TA2]6

[T1]A camada de transporte

A camada de transporte não é simplesmente outra camada. Ela é o núcleo de toda a hierarquia de protocolos. Sua função é promover uma transferência de dados confiável e econômica entre a máquina de origem e a máquina de destino, independente das redes físicas em uso no momento. Sem a camada de transporte, todo o conceito de protocolos em camadas faria pouco sentido. Neste capítulo estudaremos em detalhes a camada de transporte, bem como seus serviços, protocolos, projeto e desempenho.

[T2] 6.1 O serviço de transporte

Nas próximas seções, vamos apresentar uma introdução ao serviço de transporte. Veremos que tipo de serviço é oferecido à camada de aplicação. Para tornar mais concreta a questão do serviço de transporte, examinaremos dois conjuntos de primitivas da camada de transporte. Primeiro, estudaremos uma primitiva simples (mas hipotética) para mostrar as idéias básicas. Em seguida, analisaremos a interface comumente utilizada na Internet.

[T3] 6.1.1 Serviços oferecidos às camadas superiores

O principal objetivo da camada de transporte é oferecer um serviço confiável, eficiente e econômico a seus usuários que, em geral, são processos presentes na camada de aplicação. Para atingir esse objetivo, a camada de transporte utiliza vários serviços oferecidos pela camada de rede. O hardware/software da camada de transporte que executa o trabalho é chamado **entidade de transporte**. A entidade de transporte pode estar localizada no núcleo do sistema operacional,

em um outro processo do usuário, em um pacote de biblioteca vinculado a aplicações de rede ou na placa de interface de rede. O relacionamento (lógico) existente entre as camadas de rede, de transporte e de aplicação está ilustrado na Figura 6.1.

[arte: ver original p. 482]

[Dísticos]

[1] Host 1

Camada de aplicação (ou de sessão)

Endereço de transporte

Entidade de transporte

Endereço de rede

Camada de rede

[2] Interface aplicação/transporte

TPDU

Protocolo de transporte

Interface transporte/rede

[3] Host 2

Camada de aplicação (ou de sessão)

Entidade de transporte

Camada de rede

[F]Figura 6.1

[FL] As camadas de rede, de transporte e de aplicação

Assim como existem dois tipos de serviço de rede, o serviço orientado a conexões e o serviço sem conexões, também existem dois tipos de serviço de transporte. O serviço de transporte orientado à conexões é semelhante ao serviço de rede orientado à conexões em muitos aspectos. Em ambos os casos, as

conexões têm três fases: o estabelecimento, a transferência de dados e o encerramento. O endereçamento e o controle de fluxo também são semelhantes em ambas as camadas. Além disso, o serviço de transporte sem conexões é semelhante ao serviço de rede sem conexões.

Diante disso, fazemos a seguinte pergunta: se o serviço da camada de transporte é tão semelhante ao serviço da camada de rede, por que há duas camadas distintas? Por que uma só camada não é suficiente? A resposta, embora sutil, é de importância crucial e nos remete à Figura 1.9. O código de transporte funciona inteiramente nas máquinas dos usuários, mas a camada de rede funciona principalmente nos roteadores, cuja operação é de responsabilidade da concessionária de comunicações (pelo menos no aso de uma rede geograficamente distribuída). O que acontecerá se a camada de rede oferecer um serviço inadequado? E se perder pacotes com freqüência? O que acontecerá se os roteadores apresentarem falhas ocasionais?

É verdade que problemas acontecem. Os usuários não têm nenhum controle real sobre a camada de rede, portanto, não podem resolver o problema de um serviço ineficaz usando roteadores melhores ou aumentando o controle de erros na camada de enlace de dados. A única possibilidade é colocar sobre a camada de rede uma outra camada que melhore a qualidade do serviço. Se uma entidade de transporte for informada no meio de uma longa transmissão que sua conexão de rede foi encerrada de forma abrupta, sem nenhuma indicação do que ocorreu com os dados que estavam sendo transferidos, ela poderá gerar uma nova conexão de rede para a entidade de transporte remota. Usando essa nova conexão, ela poderá enviar uma consulta à entidade remota para saber quais dados chegaram e quais não chegaram, e depois retomar a transmissão a partir da interrupção.

Em síntese, a existência de uma camada de transporte torna o serviço de

transporte mais confiável que o serviço de rede subjacente. A ocorrência de pacotes perdidos e de dados corrompidos pode ser percebida e compensada pela camada de transporte. Além disso, as primitivas do serviço de transporte podem ser implementadas sob a forma de chamadas a procedimentos de biblioteca, a fim de torná-las independentes das primitivas do serviço de rede, que podem variar muito de rede para rede (por exemplo, um serviço de LAN sem conexões pode ser bem diferente de um serviço de WAN orientado à conexões).

Graças à camada de transporte, os programas aplicativos podem ser desenvolvidos utilizando-se um conjunto de primitivas padrão. Além disso, esses programas podem funcionar em uma ampla variedade de redes, sem a preocupação de se lidar com interfaces de sub-redes diferentes e com uma transmissão não confiável. Se todas as redes reais fossem perfeitas e se todas tivessem as mesmas primitivas de serviço e ainda a garantia de nunca mudar, provavelmente a camada de transporte não seria necessária. Porém, na vida real ela cumpre a função primordial de tornar as camadas superiores do projeto imunes à tecnologia e às imperfeições da sub-rede.

Por essa razão, muitas pessoas fazem distinção entre as camadas de 1 a 4 e as camadas acima de 4. As quatro primeiras camadas são consideradas o **provedor de serviços de transporte**, enquanto as camadas superiores constituem o **usuário de serviços de transporte.** Essa distinção entre provedor e usuário tem um impacto considerável sobre o projeto das camadas e coloca a camada de transporte em uma posição chave, pois ela representa a principal fronteira entre o provedor e o usuário do serviço de transmissão de dados confiável.

[T3] 6.1.2 Primitivas do serviço de transporte

Para permitir que os usuários tenham acesso ao serviço de transporte, a camada de transporte deve oferecer algumas operações a programas aplicativos, ou seja,

uma interface de serviço de transporte. Cada serviço de transporte tem sua própria interface. Nesta seção, veremos primeiro um serviço de transporte (hipo–tético) simples e sua interface, a fim de examinarmos os fundamentos desse tipo de serviço; na próxima seção, estudaremos um exemplo real.

O serviço de transporte é semelhante ao serviço de rede, mas existem algumas diferenças importantes entre eles. A principal delas é que o serviço de rede representa o modelo oferecido pelas redes reais, com todos os atributos. As redes reais podem perder pacotes; portanto, o serviço de rede, em geral, não é confiável.

Em contraste, o serviço de transporte (orientado a conexões) é confiável. É óbvio que as redes reais não são infalíveis, mas essa é exatamente a função da camada de transporte — oferecer um serviço confiável sobre uma rede não confiável.

Como exemplo, considere dois processos conectados por canais (pipes) no UNIX. Supomos que a conexão entre eles é perfeita, sem levar em consideração confirmações, pacotes perdidos, congestionamentos etc. Esses processos esperam ter uma conexão 100% confiável. O processo *A* insere os dados em uma extremidade do canal, e o processo *B* os recebe na outra. O transporte orientado a conexões consiste em ocultar as imperfeições do serviço de rede, de modo que os processos do usuário possam simplesmente supor a existência de um fluxo de bits livre de erros.

Por outro lado, a camada de transporte também pode oferecer um serviço não confiável (de datagramas). Porém, há relativamente pouco a dizer sobre esse assunto e assim, neste capítulo, vamos nos concentrar no serviço de transporte orientado a conexões. Apesar disso, há algumas aplicações, como a computação cliente/servidor e a multimídia de fluxo, que se beneficiam do transporte sem conexões; por essa razão, estudaremos alguns detalhes sobre ele mais adiante.

A segunda diferença entre o serviço de rede e o serviço de transporte está

relacionada ao destinatário do serviço. O serviço de rede só é usado pelas entidades de transporte. Poucos usuários criam suas próprias entidades de transporte; portanto, poucos usuários ou programas jamais vêem a estrutura do serviço de rede. Por outro lado, muitos programas (e programadores) vêem as primitivas de transporte. Por isso, o serviço de transporte deve ser adequado e fácil de usar.

Para ter uma idéia de como deve ser um serviço de transporte, considere as cinco primitivas apresentadas na Figura 6.2. Essa interface de transporte é uma estrutura básica, mas denota o que uma interface de transporte orientada a conexões deve fazer. Ela permite aos programas aplicativos estabelecer, usar e encerrar uma conexão, o que é suficiente para muitas aplicações.

[arte: ver original p. 484]

[T]Tabela

Primitiva	**Pacote enviado**	**Significado**
LISTEN	(nenhum)	Bloquear até que algum processo tente se conectar
CONNECT	CONNECTION REQ.	Tentar ativamente estabelecer uma conexão
SEND	DATA	Enviar informações
RECEIVE	(nenhum)	Bloquear até chegar um pacote DATA
DISCONNECT	DISCONNECTION REQ.	Este lado quer encerrar a conexão

[F]Figura 6.2

[FL] As primitivas para um serviço de transporte simples

Para entender como essas primitivas devem ser usadas, considere uma aplicação com um servidor e uma série de clientes remotos. Em primeiro lugar, o servidor executa uma primitiva LISTEN chamando um procedimento de biblioteca que emite uma chamada de sistema para bloquear o servidor até que um cliente

apareça. Quando um cliente quer se comunicar com o servidor, ele executa uma primitiva CONNECT. Para executar essa primitiva, a entidade de transporte bloqueia o responsável pela chamada e envia um pacote ao servidor. Encapsulada na carga útil desse pacote, encontra-se uma mensagem da camada de transporte destinada à entidade de transporte do servidor.

Faremos agora algumas observações rápidas sobre a terminologia. Por falta de um termo mais adequado, utilizamos o desajeitado acrônimo **TPDU (Transport Protocol Data Unit — unidade de dados do protocolo de transporte)** para denominar as mensagens enviadas de uma entidade de transporte a outra entidade de transporte. Portanto, as TPDUs (intercambiadas pela camada de transporte) estão contidas em pacotes (intercambiados pela camada de rede). Por sua vez, os pacotes estão contidos em quadros (intercambiados pela camada de enlace de dados). Quando um quadro chega, a camada de enlace de dados processa o cabeçalho do quadro e transmite o conteúdo do campo de carga útil do quadro à entidade de rede. Em seguida, a entidade de rede processa o cabeçalho do pacote e envia o conteúdo da carga útil do pacote à entidade de transporte. Esse aninhamento é ilustrado na Figura 6.3.

[arte: ver original p. 485]

[Dísticos]

[1] Cabeçalho do quadro

[2] Cabeçalho do pacote

[3] Cabeçalho da TPDU

[4] Carga útil da TPDU

[5] Carga útil do pacote

[6] Carga útil do quadro

[F]Figura 6.3

[FL] Aninhamento de TPDUs, pacotes e quadros

Voltando ao exemplo de cliente/servidor, a chamada CONNECT do cliente faz a TPDU CONNECTION REQUEST ser enviada ao servidor. Quando a TPDU chega ao servidor, a entidade de transporte verifica se o servidor está bloqueado em uma primitiva LISTEN (ou seja, apto a administrar solicitações). Em seguida, a TPDU desbloqueia o servidor e transmite uma TPDU CONNECTION ACCEPTED de volta para o cliente. Quando essa TPDU chega a seu destino, o cliente é desbloqueado e a conexão é estabelecida.

A partir desse momento é possível intercambiar dados utilizando-se as primitivas SEND e RECEIVE. Em sua forma mais simples, qualquer uma das partes pode executar uma primitiva RECEIVE (bloqueio) para aguardar que a outra execute uma primitiva SEND. Quando a TPDU chega a seu destino, o receptor é desbloqueado. Em seguida, ele pode processar a TPDU e enviar uma resposta. Esse sistema funciona bem desde que as partes controlem de quem é a vez de enviar os dados.

Cabe observar que, na camada de transporte, até mesmo uma troca de dados simples e unidirecional é mais complexa do que na camada de rede. Cada pacote de dados enviado também será confirmado (eventualmente). Os pacotes que transportam TPDUs de controle também são confirmados, de forma explícita ou implícita. Essas confirmações são gerenciadas por entidades de transporte que utilizam o protocolo da camada de rede e não são visíveis para os usuários de transporte. Da mesma forma, as entidades de transporte têm de lidar com timers e retransmissões. Esse mecanismo também não é percebido pelos usuários de transporte. Para eles, uma conexão é um canal de bits confiável: um usuário envia bits por uma das extremidades e eles são percebidos na outra como em um passe de mágica. A habilidade de ocultar sua complexidade é o que torna os protocolos em camadas uma ferramentas tão útil.

Quando uma conexão não é mais necessária, ela deve ser encerrada para desocupar espaço de tabela dentro das duas entidades de transporte. O encerramento da conexão tem duas variantes: assimétrica e simétrica. Na variante assimétrica, qualquer um dos usuários de transporte pode emitir uma primitiva DISCONNECT, fazendo com que uma TPDU DISCONNECT seja enviada à entidade de transporte remota. Quando essa TPDU chega a seu destino, a conexão é encerrada.

Na variante simétrica, cada direção da comunicação é encerrada separadamente e de forma independente da outra. Quando uma das partes executa uma primitiva DISCONNECT, isso significa que não há mais dados a enviar, mas ainda é possível receber dados enviados pela outra parte. Nesse modelo, a conexão só é encerrada quando os dois lados tiverem executado uma primitiva DISCONNECT.

Um diagrama de estados para o estabelecimento e encerramento de uma conexão com essas primitivas simples é mostrado na Figura 6.4. Cada transição é acionada por algum evento, seja ele uma primitiva executada pelo usuário de transporte local ou um pacote que chega. Por simplicidade, supomos que cada TPDU é confirmada separadamente e que um modelo de desconexão simétrica está sendo usado, com o cliente dando início ao procedimento. Cabe observar que esse modelo é muito comum. Veremos adiante outros modelos mais realistas.

[arte: ver original p. 486]

[Dísticos]

[1] *TPDU Connection request recebida*

ESTABELECIMENTO PASSIVO PENDENTE

[2] OCIOSA Primitiva Connect executada

ESTABELECIMENTO ATIVO PENDENTE

[3] Primitiva Connet executada ESTABELECIDA *TPDU Connection*

accepted recebida

TPDU Disconnection request recebida Primitiva Disconnect executada

[4] DESCONEXÃO PASSIVA PENDENTE DESCONEXÃO ATIVA PENDENTE

[5] Primitiva Disconnect executada OCIOSA *TPDU Disconnection Request recebida*

[F]Figura 6.4

[FL] Diagrama de estados para um esquema simples de gerenciamento de conexão. As transições identificadas em itálico são causadas pela chegada de pacotes. As linhas contínuas mostram a seqüência de estados do cliente. As linhas tracejadas mostram a seqüência de estados do servidor

[T3] 6.1.3 Soquetes de Berkeley

Agora, vamos analisar resumidamente outro conjunto de primitivas de transporte, as primitivas de soquetes usadas no UNIX de Berkeley para o TCP. Essas primitivas são mostradas na Figura 6.5 e são amplamente usadas em programação para a Internet. Grosso modo, elas seguem o mesmo modelo do nosso primeiro exemplo, mas oferecem mais recursos e maior flexibilidade. Não vamos mostrar as TPDUs correspondentes a elas nesta seção; essa discussão ficará para mais adiante, quando estudarmos o TCP.

As quatro primeiras primitivas na lista são executadas pelos servidores nessa mesma ordem. A primitiva SOCKET cria um novo ponto final e aloca espaço de tabela para ele na entidade de transporte. Os parâmetros da chamada especificam o formato de endereçamento a ser usado, o tipo de serviço desejado (por exemplo, um fluxo de bytes confiável) e o protocolo. Uma chamada SOCKET bem-sucedida retorna um descritor de arquivo comum que será usado nas chamadas subseqüentes, exatamente como uma chamada OPEN.

[arte: ver original p. 487]

[T]Tabela

Primitiva	**Significado**
SOCKET	Criar um novo ponto final de comunicação
BIND	Anexar um endereço local a um soquete
LISTEN	Anunciar a disposição para aceitar conexões; mostra o tamanho da fila
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida
CONNECT	Tentar estabelecer uma conexão ativamente
SEND	Enviar alguns dados através da conexão
RECEIVE	Receber alguns dados da conexão
CLOSE	Encerrar a conexão

[F]Figura 6.5

[FL] As primitivas de soquetes para TCP

Os soquetes recém-criados não têm endereços de rede; esses endereços são atribuídos através de uma primitiva BIND. Uma vez que um servidor tenha designado um endereço para um soquete, os clientes remotos já podem se conectar a ele. A razão para uma chamada SOCKET não criar um endereço diretamente é que alguns processos consideram importante manter seus endereços (por exemplo, eles vêm usando o mesmo endereço há muitos anos e todos já o conhecem), ao passo que outros não se importam com isso.

Em seguida, temos a chamada LISTEN, que aloca espaço para a fila de chamadas recebidas, caso vários clientes tentem se conectar ao mesmo tempo. Ao contrário da chamada LISTEN em nosso primeiro exemplo, a chamada LISTEN do modelo de soquetes não é uma chamada de bloqueio.

Para bloquear a espera por uma conexão de entrada, o servidor executa uma

primitiva ACCEPT. Quando chega uma TPDU solicitando uma conexão, a entidade de transporte cria um novo soquete com as mesmas propriedades do soquete original e retorna um descritor de arquivo para ele. Em seguida, o servidor pode desviar um processo ou um thread para tratar a conexão no novo soquete e voltar a esperar pela próxima conexão no soquete original. ACCEPT retorna um descritor de arquivo normal, que pode ser usado para ler e gravar da maneira padrão, como no caso de arquivos.

Agora, vamos ver o que acontece no lado cliente. Aqui também é preciso criar primeiro um soquete usando a primitiva SOCKET, mas a primitiva BIND não é necessária, pois o endereço usado não é importante para o servidor. A primitiva CONNECT bloqueia o responsável pela chamada e inicia o processo de conexão. Quando a conexão é concluída (ou seja, quando a TPDU apropriada é recebida do servidor), o processo cliente é desbloqueado e a conexão é estabelecida. Depois disso, ambos os lados podem usar as primitivas SEND e RECV para transmitir e receber dados através da conexão full-duplex.

O encerramento da conexão com soquetes é simétrico. Quando ambos os lados tiverem executado uma primitiva CLOSE, a conexão será encerrada.

[T3] 6.1.4 Um exemplo de programação de soquetes: um servidor de arquivos da Internet

Como um exemplo de utilização das chamadas de soquetes, considere o código do cliente e o código do servidor da Figura 6.6. Nessa figura, temos um servidor de arquivos da Internet muito primitivo, juntamente com um exemplo de cliente que o utiliza. O código tem muitas limitações (descritas a seguir) mas, em princípio, o código do servidor pode ser compilado e executado em qualquer sistema UNIX conectado à Internet. O código do cliente pode então ser compilado e executado em qualquer outra máquina UNIX conectada à Internet, situada em

qualquer lugar do mundo. O código do cliente pode ser executado com parâmetros apropriados para buscar qualquer arquivo ao qual o servidor tem acesso em sua máquina. O arquivo é gravado na saída padrão que, é claro, pode ser redirecionada para um arquivo ou um canal.

Vamos examinar primeiro o código do servidor. Ele começa incluindo alguns cabeçalhos padrão, e os três últimos cabeçalhos contêm as principais definições e estruturas de dados relacionadas à Internet. Em seguida, temos uma definição de *SERVER_PORT* como 12345. Esse número foi escolhido arbitrariamente. Qualquer número entre 1024 e 65535 também funcionará, desde que não esteja em uso por algum outro processo. É claro que o cliente e o servidor têm de usar a mesma porta. Se esse servidor se tornar um sucesso mundial (algo improvável, dado seu caráter primitivo), será atribuída a ele uma porta permanente abaixo de 1024, e aparecerá em *www.iana.org*.

As duas linhas seguintes no servidor definem duas constantes necessárias. A primeira define o tamanho do bloco usado na transferência de arquivos. A segunda determina quantas conexões pendentes podem ser mantidas antes de serem descartadas conexões adicionais que chegarem.

Depois das declarações de variáveis locais, tem início o código do servidor. Ele começa inicializando uma estrutura de dados que conterá o endereço IP do servidor. Essa estrutura de dados logo será vinculada ao soquete do servidor. A chamada a *memset* define a estrutura de dados como somente valores 0. As três atribuições seguintes preenchem três de seus campos. A última dessas atribuições contém a porta do servidor. As funções *htonl* e *htons* estão relacionadas com a conversão de valores para um formato padrão, de modo que o código possa funcionar corretamente em máquinas big-endian (por exemplo, o SPARC) e em máquinas little-endian (por exemplo, o Pentium). Sua semântica exata não é relevante no momento.

Em seguida, o servidor cria um soquete e verifica se ele contém erros (indicados por $s < 0$). Em uma versão de produção do código, a mensagem de erro poderia ser um pouco mais explicativa. A chamada a *setsockopt* é necessária para permitir que a porta seja reutilizada, de forma que o servidor possa funcionar indefinidamente, recebendo solicitação após solicitação. Agora, o endereço IP é vinculado ao soquete e é realizada uma verificação para saber se a chamada a *bind* teve sucesso. A última etapa da inicialização é a chamada a *listen* para anunciar a disposição do servidor para aceitar chamadas de entrada, e informar ao sistema que ele deve armazenar um número de chamadas igual a *QUEUE_SIZE*, no caso de chegarem novas solicitações enquanto o servidor ainda estiver processando a solicitação atual. Se a fila estiver cheia e chegarem solicitações adicionais, elas serão simplesmente descartadas.

Nesse momento, o servidor entra em seu loop principal, do qual ele nunca sai. A única maneira de interromper o loop é eliminá-lo do exterior. A chamada a *accept* bloqueia o servidor até algum cliente tentar estabelecer uma conexão com ele. Se a chamada *accept* tiver êxito, ela retornará um descritor de arquivo que poderá ser usado para leitura e gravação, de maneira análoga ao uso de descritores de arquivos em operações de leitura e gravação de canais. Porém, diferentes dos canais, que são unidirecionais, os soquetes são bidirecionais, e assim o endereço de soquete (*sa* — socket address) pode ser usado para realizar a leitura da conexão e também para efetuar gravações.

Depois que a conexão é estabelecida, o servidor lê o nome do arquivo. Se o nome ainda não estiver disponível, o servidor será bloqueado esperando por ele. Após obter o nome do arquivo, o servidor abre o arquivo e, em seguida, entra em um loop que, alternadamente, lê blocos do arquivo e os grava no soquete, até copiar o arquivo inteiro. Depois, o servidor fecha o arquivo e a conexão, e espera que a próxima conexão apareça. Ele repete esse loop para sempre.

Agora vamos examinar o lado cliente. Para entender como ele funciona, é necessário compreender como ele é invocado. Supondo-se que ele seja denominado *client*, uma chamada típica será:

[TD] client flits.cs.vu.nl /usr/tom/nomearq >f [TN]

Essa chamada só funciona se o servidor já estiver em execução em *flits.cs.vu.nl* e o arquivo */usr/tom/nomearq* existir, e se o servidor tiver acesso de leitura para o arquivo. Se a chamada for bem sucedida, o arquivo será transferido pela Internet e gravado em *f*; depois disso, o programa cliente será encerrado. Tendo em vista que o servidor continua após uma transferência, o cliente pode ser iniciado repetidamente para obter outros arquivos.

O código do cliente começa com algumas inclusões e declarações. A execução se inicia verificando-se se o código foi chamado com o número correto de argumentos (*argc* = 3 representa o nome do programa e mais dois argumentos). Observe que *argv* [1] contém o nome do servidor (por exemplo, *flits.cs.vu.nl*) e é convertido em um endereço IP por *gethostbyname*. Essa função utiliza o DNS para pesquisar o nome. Vamos estudar o DNS no Capítulo 7.

Em seguida, um soquete é criado e inicializado. Depois disso, o cliente tenta estabelecer uma conexão do TCP para o servidor, usando *connect*. Se o servidor estiver funcionando na máquina nomeada e conectado a *SERVER_PORT*, e se ele estiver ocioso ou tiver espaço em sua fila *listen*, a conexão será (eventualmente) estabelecida. Usando a conexão, o cliente envia o nome do arquivo gravando-o no soquete. O número de bytes enviados é uma unidade maior que o nome em si, tendo em vista que o byte 0 que encerra o nome também deve ser enviado para informar ao servidor em que ponto o nome termina.

[arte: ver original da p. 490-491]

[TD]

/* Esta página contém um programa cliente que pode solicitar um arquivo

do programa servidor representado na próxima página. O servidor responde enviando o arquivo inteiro.

*/

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345              /* arbitrário, mas o cliente e o servidor
                                       /* devem concordar */
#define BUF_SIZE 4096                  /* tamanho de transferência de bloco */

int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                  /* buffer para arquivo recebido */
  struct hostent *h;                   /* info sobre servidor */
  struct sockaddr_in channel;          /* contém endereço IP */

  if (argc != 3) fatal("Usage: client server-name file-name");
  h = gethostbyname(argv[1]);          /* pesquisa endereço IP do host */
  if (!h) fatal("gethostbyname failed");

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s < 0) fatal("socket");
  memset(&channel, 0, sizeof(channel));
```

```
  channel.sin_family= AF_INET;

  memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);

  channel.sin_port= htons(SERVER_PORT);

  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));

  if (c < 0) fatal("connect failed");


  /* Agora a conexão é estabelecida. Envia nome do arquivo incluindo o byte 0 no
final. */

  write(s, argv[2], strlen(argv[2])+1);


  /* Recebe o arquivo e o grava na saída padrão . */

  while (1) {

        bytes = read(s, buf, BUF_SIZE);    /* lê do soquete */

        if (bytes <= 0) exit(0);       /* verifica fim de arquivo */

        write(1, buf, bytes);          /* grava na saída padrão */

  }

}


fatal(char *string)

{

  printf("%s\n", string);

  exit(1);

} [TN]
```

[F]Figura 6.6

[FL] Código do cliente utilizando soquetes. O código do servidor encontra-se na próxima página [Atenção produção!]

[TD]

```
#include <sys/types.h>                /* Este é o código do servidor */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345             /* arbitrário, mas cliente e servidor devem
concordar */
#define BUF_SIZE 4096                 /* tamanho de transferência de bloco */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                 /* buffer para arquivo de saída */
  struct sockaddr_in channel;         /* contém endereço IP */

  /* Constrói estrutura de endereço para vincular ao soquete. */
  memset(&channel, 0, sizeof(channel));       /* canal zero */
  channel.sin_family = AF_INET;
  channel.sin_addr.s_addr = htonl(INADDR_ANY);
  channel.sin_port = htons(SERVER_PORT);

  /* Abertura passiva. Espera por conexão. */
  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); /* cria soquete */
  if (s < 0) fatal("socket failed");
```

```
  setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
  if (b < 0) fatal("bind failed");

  l = listen(s, QUEUE_SIZE);              /* especifica tamanho de fila */
  if (l < 0) fatal("listen failed");

  /* Agora o soquete está configurado e vinculado. Espera por conexão e a
processa. */
  while (1) {
        sa = accept(s, 0, 0);        /* solicitação de bloco para conexão */
        if (sa < 0) fatal("accept failed");

        read(sa, buf, BUF_SIZE);     /* lê nome de arquivo em soquete */

        /* Recebe e retorna o arquivo. */
        fd = open(buf, O_RDONLY);        /* abre o arquivo para ser enviado de volta
*/
        if (fd < 0) fatal("open failed");

        while (1) {
              bytes = read(fd, buf, BUF_SIZE); /* lê o arquivo */
              if (bytes <= 0) break;           /* verifica fim de arquivo */
              write(sa, buf, bytes);           /* grava bytes no soquete */
        }
        close(fd);                             /* fecha arquivo */
```

```
      close(sa);                          /* fecha conexão */
  }
}

fatal(char *string)
{
  printf("%s", string);
  exit(1);
} [TN]
```

Agora o cliente entra em um loop, lendo o arquivo bloco por bloco do soquete e copiando-o na saída padrão. Ao terminar, ele simplesmente encerra o loop.

O procedimento *fatal* imprime uma mensagem de erro e encerra. O servidor precisa do mesmo procedimento, mas ele foi omitido por falta de espaço na página. Tendo em vista que o cliente e o servidor são compilados separadamente e em geral funcionam em computadores diferentes, eles não podem compartilhar o código de *fatal*.

Esses dois programas (bem como outros materiais relacionados a este livro) podem ser encontrados no Web site do livro em:

[TD] http://www.prenhall.com/tanenbaum [TN]

ao se clicar no link Web Site ao lado da foto da capa. Eles podem ser baixados e compilados em qualquer sistema UNIX (por exemplo, Solaris, BSD, Linux), usando:

[TD]

```
cc -o client client.c -lsocket -lnsl
cc -o server server.c -lsocket -lnsl [TN]
```

O servidor é iniciado digitando-se:

[TD] server [TN]

O cliente precisa de dois argumentos, conforme descrevemos antes. Também está disponível no Web site uma versão para Windows.

Apenas para registrar, esse servidor não é a última palavra em desempenho. Sua verificação de erros é escassa e seu relatório de erros é medíocre. É claro que ele nunca ouviu falar de segurança, e usar chamadas comuns do sistema UNIX não é a última palavra em independência da plataforma. Ele também faz algumas suposições tecnicamente ilegais, como a de considerar que o nome de arquivo cabe no buffer e é transmitido de forma atômica. Como ele trata todas as solicitações de modo estritamente seqüencial (porque tem apenas um thread), seu desempenho é fraco. Apesar dessas limitações, é um servidor de arquivos da Internet completo e funcional. Nos exercícios, o leitor será convidado a aperfeiçoá-lo. Para obter mais informações sobre programação com soquetes, consulte (Stevens, 1997).

[T2] 6.2 Elementos de protocolos de transporte

O serviço de transporte é implementado por um **protocolo de transporte** usado entre duas entidades de transporte. Em alguns aspectos, os protocolos de transporte lembram os protocolos de enlace de dados que estudamos em detalhes no Capítulo 3. Ambos têm de lidar com o controle de erros, com a definição de seqüências e com o controle de fluxo, entre outros itens. Entretanto, existem diferenças significativas entre os dois. Essas diferenças ocorrem devido às peculiaridades dos ambientes nos quais os dois protocolos operam, como mostra a Figura 6.7. Na camada de enlace de dados, dois roteadores se comunicam diretamente através de um canal físico, enquanto na camada de transporte esse canal físico é substituído pela sub-rede inteira. Essa diferença tem muitas implicações importantes para os protocolos, como veremos neste capítulo.

[arte: ver original p. 493]

[Dísticos]

[1] Roteador

[2] Canal de comunicação física

(a)

[3] Roteador sub-rede

[4] Host

(b)

[F]Figura 6.7

[FL] (a)Ambiente da camada de enlace de dados. (b)Ambiente da camada de transporte

Em primeiro lugar, na camada de enlace de dados o roteador não precisa especificar com que roteador deseja se comunicar, pois cada linha de saída especifica de modo exclusivo um determinado roteador. Na camada de transporte, é necessário o endereçamento explícito de destinos.

Por outro lado, o processo de estabelecimento de uma conexão através de um cabo, como mostra a Figura 6.7(a), é simples: a outra extremidade está sempre presente (a menos que haja alguma falha). De qualquer modo, não há muito a fazer. Já na camada de transporte, o estabelecimento inicial da conexão é mais complicado, como veremos mais adiante.

Outra diferença, extremamente inoportuna, entre a camada de enlace de dados e a camada de transporte é a possível existência de capacidade de armazenamento na sub-rede. Quando um roteador envia um quadro, ele pode chegar a seu destino ou se perder, mas não ricocheteia nem se esconde em algum canto para emergir de repente, em um momento inoportuno, 30 segundos depois. Se a sub-rede utilizar datagramas e um roteamento adaptativo interno, haverá uma

probabilidade significativa de que um pacote possa ficar armazenado por alguns segundos e ser entregue mais tarde. As conseqüências da habilidade da sub-rede para armazenar pacotes podem às vezes ser desastrosas e exigir o uso de protocolos especiais.

A última diferença entre as camadas de enlace de dados e de transporte é mais uma questão de volume, e não de espécie. Os controles de buffers e de fluxo são necessários em ambas as camadas, mas a presença de um número grande e dinamicamente variável de conexões na camada de transporte pode exigir outra estratégia que não a da camada de enlace de dados. No Capítulo 3, vimos que alguns dos protocolos alocam um número fixo de buffers para cada linha. Portanto, quando um quadro chegar, sempre haverá um buffer disponível. Na camada de transporte, o grande número de conexões que precisam ser gerenciadas torna menos atraente a idéia de dedicar vários buffers a cada uma. Nas próximas seções, vamos analisar essas e outras questões importante.

[T3] 6.2.1 Endereçamento

Quando um processo de aplicação (por exemplo, um usuário) deseja estabelecer uma conexão com um processo de aplicação remoto, é necessário especificar a aplicação com a qual ele irá se conectar. (O transporte sem conexões tem o mesmo problema: a quem cada mensagem deve ser enviada?) O método normalmente utilizado é definir os endereços de transporte que os processos podem ouvir para receber solicitações de conexão. Na Internet, essas extremidades são chamadas **portas.** Em redes ATM, elas se denominam AAL-SAPs. Vamos utilizar o acrônimo **TSAP (Transport Service Access Point — ponto de acesso de serviço de transporte).** Os pontos extremos análogos na camada de rede (ou seja, os endereços da camada de rede) são chamados então **NSAPs.** Os endereços IP são exemplos de NSAPs.

A Figura 6.8 ilustra o relacionamento entre o NSAP, o TSAP e a conexão de transporte. Os processos de aplicações, tanto clientes quanto servidores, podem se associar a um TSAP para estabelecer uma conexão com um TSAP remoto. Essas conexões funcionam através dos NSAPs de cada host, como mostra a figura. O propósito de ter TSAPs é o fato de, em algumas redes, cada computador ter um único NSAP, e assim necessitar de algum meio para distinguir entre vários pontos extremos de transporte que compartilham esse NSAP.

[arte: ver original p. 494]

[Dísticos]

[1]Host 1

Processo de aplicação TSAP 1208

Conexão de transporte

NSAP

[2]Camada de aplicação

Camada de transporte

Camada de rede

Camada de enlace de dados

Camada física

[3]Host 2

Servidor 1 Servidor 2

TSAP 1522 TSAP 1836

NSAP

[F]Figura 6.8

[FL] TSAPs, NSAPs e conexões de transporte

Aqui está um possível cenário para uma conexão de transporte:

1. Um processo servidor de hora do dia no host 2 se associa ao TSAP 1522 para

aguardar a chegada de uma chamada. O modo como um processo se associa a um TSAP está fora do escopo do modelo de rede e depende exclusivamente do sistema operacional local. Por exemplo, poderia ser utilizada uma chamada como a nossa primitiva LISTEN.

2. Um processo de aplicação no host 1 deseja descobrir a hora do dia e, portanto, transmite uma solicitação CONNECT especificando o TSAP 1208 como origem e o TSAP 1522 como destino. Em última análise, essa ação resulta no estabelecimento de uma conexão de transporte entre o processo de aplicação do host 1 e o servidor 1 do host 2.
3. O processo de aplicação envia então uma solicitação para saber a hora.
4. O processo servidor de hora responde com a hora atual.
5. A conexão de transporte é então encerrada (liberada).

Observe que pode haver outros servidores no host 2, associados a outros TSAPs e que estejam esperando pela chegada de conexões de entrada sobre o mesmo NSAP.

O quadro mostrado anteriormente é bom, exceto pelo fato de termos omitido um pequeno problema: como o processo de usuário do host 1 sabe que o servidor da hora do dia está associado ao TSAP 1522? Uma possibilidade é o servidor de hora do dia estar associado ao TSAP 1522 há anos e, gradualmente, todos os usuários da rede terem se habitado com isso. Nesse modelo, os serviços têm endereços TSAP estáveis que estão listados em arquivos guardados em locais conhecidos, como o arquivo */etc/services* dos sistemas UNIX, que lista os servidores que estão associados de modo permanente a cada uma das portas.

Enquanto os endereços TSAP estáveis podem funcionar para um pequeno número de serviços que nunca mudam (por exemplo, o servidor da Web), em geral, os processos do usuário desejam se comunicar com outros processos do usuário que existem por um curto período de tempo e, portanto, não têm um endereço

TSAP previamente conhecido. Além disso, se houver potencialmente muitos processos servidores, muitos dos quais raramente usados, é um desperdício mantê-los ativos e na escuta de um endereço TSAP estável durante um dia inteiro. E necessário um esquema melhor.

Tal esquema, usado por hosts UNIX na Internet, é ilustrado de maneira simplificada na Figura 6.9. Ele é conhecido como **protocolo de conexão inicial**. Em vez de ter todos os servidores associados a um TSAP conhecido, cada máquina que desejar oferecer serviços a usuários remotos terá um **servidor de processos** especial que funcionará como um proxy para servidores menos utilizados. Ele atende a uma série de portas ao mesmo tempo, aguardando uma solicitação de conexão. Os usuários potenciais de um serviço devem começar com uma solicitação CONNECT, especificando o endereço TSAP do serviço que desejam. Se nenhum servidor os estiver aguardando, eles estabelecem uma conexão com o servidor de processos, como é mostra a Figura 6.9(a).

Depois de receber a solicitação, o servidor de processos gera a conexão para o servidor solicitado, permitindo que ele herde a conexão já existente com o usuário. Em seguida, o novo servidor executa a tarefa solicitada, enquanto o servidor de processos volta a aguardar novas solicitações, como mostra a Figura 6.9(b).

Enquanto o protocolo de conexão inicial funciona bem com os servidores que podem ser criados quando necessário, há muitas situações em que os serviços existem de forma independente do servidor de processos. Por exemplo, um servidor de arquivos deve ser executado em um hardware especial (uma máquina com uma unidade de disco) e não pode ser criado dinamicamente quando alguém deseja se comunicar com ele.

[arte: ver original p. 496]

[Dísticos]

[1]Host 1 Host 2

Camada

Usuário Servidor de processos

4 TSAP

(a)

[2]Host 1 Host 2

Servidor de hora do dia

Usuário Servidor de processos

(b)

[F]Figura 6.9

[FL] Como um processo do usuário no host 1 estabelece uma conexão com o servidor de hora do dia no host 2

Com freqüência, é usado um esquema alternativo para administrar essa situação. Nesse modelo existe um processo especial chamado **servidor de nomes** ou, às vezes, **servidor de diretórios.** Para localizar o endereço do TSAP correspondente a um determinado nome de serviço como, por exemplo "hora do dia", um usuário estabelece uma conexão com o servidor de nomes (que está associado a um TSAP conhecido). Em seguida, o usuário envia uma mensagem especificando o nome do serviço, e o servidor de nomes retorna o endereço do TSAP. Depois disso, o usuário encerra a conexão com o servidor de nomes e estabelece uma nova conexão com o serviço desejado.

Nesse modelo, quando um novo serviço é criado, ele deve se registrar no servidor de nomes, fornecendo seu nome de serviço (normalmente um string ASCII) e o endereço de seu TSAP. O servidor de nomes registra essas informações em seu banco de dados interno, a fim de poder fornecer as respostas quando houver consultas mais tarde.

A função do servidor de nomes é análoga à da telefonista de auxílio à lista — ele fornece o mapeamento de nomes relacionados a números. Assim como ocorre no sistema telefônico, é essencial que o endereço do TSAP conhecido, usado pelo servidor de nomes (ou pelo servidor de processos no protocolo da conexão inicial), seja de fato bem conhecido. Se o número da telefonista de auxílio à lista não for conhecido, a consulta não poderá ser feita. Se você acha que o número para informações é óbvio, experimente usá-lo em outro país.

[T3] 6.2.2 Estabelecimento de conexões

Estabelecer uma conexão parece uma tarefa fácil mas, na verdade, trata-se de um procedimento complicado. À primeira vista, pode parecer que basta uma entidade de transporte enviar uma TPDU CONNECTION REQUEST ao destino e aguardar uma resposta CONNECTION ACCEPTED. O problema é que a rede pode perder, armazenar e duplicar pacotes. Esse comportamento causa sérias complicações. Imagine uma sub-rede tão congestionada que as confirmações quase nunca chegam a tempo, cada pacote sofre um timeout e é retransmitido duas ou três vezes. Suponha que a sub-rede utilize datagramas internamente e que cada pacote siga uma rota específica. Alguns pacotes podem ficar detidos na sub-rede e demorar muito para chegar, ou seja, eles ficam armazenados na sub-rede e só surgem muito depois.

O pior que pode acontecer é um usuário estabelecer uma conexão com um banco, enviar mensagens dizendo ao banco para transferir uma grande quantia para a conta de uma pessoa não muito confiável, e depois encerrar a conexão. Para seu infortúnio, nesse cenário, cada pacote é duplicado e armazenado na sub-rede. Depois da conexão ter sido encerrada, todos os pacotes surgem da sub-rede e chegam ao destino em ordem, solicitando que o banco estabeleça uma nova conexão, transfira dinheiro (de novo) e encerre a conexão. O banco não

tem como saber que se trata de uma duplicata do pedido. Ele supõe que essa é uma segunda transação independente, e transfere o dinheiro mais uma vez. No restante desta seção, vamos estudar o problema das duplicatas atrasadas, dando ênfase especial aos algoritmos para estabelecer conexões confiáveis, de modo a evitar problemas como o que acabamos de descrever.

O ponto crucial do problema é a existência de duplicatas atrasadas. Ele pode ser combatido de várias maneiras; no entanto, nenhuma delas é muito satisfatória. Uma alternativa possível é usar endereços de transporte descartáveis. Nessa concepção, cada vez que um endereço de transporte é necessário, um novo endereço é criado. Ao fim da conexão, o endereço é descartado e nunca mais é usado novamente. Essa estratégia inviabiliza o modelo de servidor de processos da Figura 6.9.

Outra possibilidade é atribuir a cada conexão um identificador de conexão (isto é, um número de seqüência incrementado a cada conexão estabelecida), escolhido pelo lado que inicia a conexão e colocado em cada TPDU, inclusive naquela que solicita a conexão. Após o encerramento das conexões, cada entidade de transporte pode atualizar uma tabela que lista as conexões obsoletas como pares (entidade de transporte correspondente, identificador de conexão). Sempre que uma solicitação de conexão for recebida, será possível compará-la com a tabela para verificar se ela pertencia a uma conexão já encerrada.

Infelizmente, esse esquema tem uma falha básica: ele exige que cada entidade de transporte mantenha uma certa quantidade de informações sobre o histórico das conexões por um tempo indeterminado. Se uma máquina quebrar e perder sua memória, ela não terá mais como saber quais identificadores de conexões já foram utilizados.

É necessário usar outro método. Em vez de permitir que os pacotes permaneçam na sub-rede eternamente, devemos criar um mecanismo para destruir os pacotes

desatualizados que ainda estejam presentes. Se estipularmos um tempo máximo de duração para um pacote, o problema poderá ser contornado.

A duração de um pacote pode ser limitada a uma valor máximo conhecido, usando-se uma (ou mais) destas técnicas:

1. Restringir o projeto da sub-rede.
2. Usar um contador de hops em cada pacote.
3. Utilizar um timbre de hora em cada pacote.

O primeiro método inclui qualquer procedimento que evite que um pacote execute um loop, combinado com algum meio de limitar o retardo devido ao congestionamento sobre o caminho mais longo possível (agora conhecido). O segundo método consiste em ter um contador de hops, inicializado com algum valor apropriado e decrementado toda vez que o pacote é encaminhado. O protocolo de rede simplesmente descarta qualquer pacote cujo contador de hops chega a zero. O terceiro método exige que cada pacote seja associado ao horário em que foi criado, com a concordância dos roteadores em descartar qualquer pacote mais antigo que um horário previamente estabelecido. Esse último método exige que os relógios dos roteadores estejam sincronizados, o que não é uma tarefa fácil, a menos que a sincronização seja obtida fora da rede, por exemplo, usando um GPS ou alguma estação de rádio que transmita periodicamente a hora exata.

Na prática, é necessário garantir que não apenas o pacote foi destruído, mas também que todas as suas confirmações também foram destruídas; portanto, introduziremos agora o valor T, algum múltiplo pequeno da verdadeira duração máxima do pacote. O múltiplo depende de um protocolo e tem o efeito de tornar T mais longo. Se esperarmos um tempo T após um pacote ser enviado, poderemos ter certeza de que todos os traços do pacote desapareceram, e que nem ele nem suas confirmações surgirão repentinamente para complicar nosso

trabalho.

Limitando o tempo de vida dos pacotes, é possível imaginar uma forma infalível de estabelecer conexões com segurança. O método descrito a seguir foi proposto por Tomlinson (1975); ele resolve o problema, mas introduz algumas peculiaridades. Esse mesmo método foi aprimorado mais tarde por Sunshine e Dalal (1978). Na prática, muitas de suas variações são amplamente utilizadas, inclusive no TCP.

Para contornar o problema da perda de memória de uma máquina após um desastre, Tomlinson propôs que cada host fosse equipado com um relógio. Os relógios dos hosts não precisam estar sincronizados. Cada relógio assume a forma de um contador binário incrementado em intervalos regulares. Além disso, o número de bits no contador deve ser maior ou igual ao número de bits dos números de seqüência. Por último, e mais importante, o relógio deve continuar funcionando mesmo que o host saia do ar.

A idéia básica é assegurar que duas TPDUs com números idênticos jamais fiquem pendentes ao mesmo tempo. Quando uma conexão é estabelecida, os k bits de baixa ordem do relógio são usados como o número de seqüência inicial (também de k bits). Assim, diferente de nossos protocolos do Capítulo 3, cada conexão começa a numerar suas TPDUs com um número de seqüência inicial diferente. O espaço de seqüência deve ser tão extenso que, quando os números de seqüência começarem a se repetir, as TPDUs antigas com o mesmo número de seqüência já deverão ter sido destruídas há muito tempo. O relacionamento linear entre o tempo e o número de seqüência é ilustrado na Figura 6.10.

Uma vez que duas entidades de transporte tenham chegado a um acordo sobre o número de seqüência inicial, qualquer protocolo de janela deslizante poderá ser usado para o controle de fluxo de dados. Na verdade, a curva do número de seqüência inicial (representada pela linha mais grossa) não é exatamente uma

reta, e se parece com uma escada, pois o relógio avança em passos discretos. Para simplificar, vamos ignorar esse detalhe.

Ocorre um problema quando um host sofre uma pane. Quando ele retorna à atividade, sua entidade de transporte não sabe onde ele estava no espaço de seqüência. Uma solução possível é determinar que as entidades de transporte fiquem inativas durante *T* segundos após uma recuperação, a fim de permitir que todas as TPDUs antigas sejam eliminadas. No entanto, em uma inter-rede complexa, o intervalo *T* pode ser muito grande e assim essa estratégia não é a mais indicada.

[arte: ver original p. 499]

[Dísticos]

[1]Números de seqüência

120

80

70

60

0

[2]Mensagem proibida T

Região proibida

Reinício após desastre com 70

[3]0 30 60 90 120 150 180

Tempo

(a)

[4]Números de seqüência

$2^k - 1$

[5] T

Números de seqüência reais utilizados

[6]Tempo

(b)

[F]Figura 6.10

[FL] (a)As TPDUs não podem entrar na região proibida. (b) O problema da ressincronização

Para evitar que seja necessário um tempo de inatividade de *T* segundos após uma falha, temos de incluir uma nova restrição ao uso dos números de seqüência. A maneira mais fácil de entender a necessidade dessa restrição é analisar um exemplo. Suponhamos que *T*, o tempo máximo de duração de um pacote, seja igual a 60 segundos e que o relógio pulse em intervalos de um segundo. Conforme mostra a linha mais grossa da Figura 6.10(a), o número de seqüência inicial para uma conexão aberta no momento *x* será *x*. Imagine que, quando t = 30 segundos, uma TPDU de dados comum que está sendo enviada através de uma conexão 5 (previamente estabelecida) recebe o número de seqüência 80. Chame essa TPDU de *X*. Logo após ter enviado a TPDU *X*, o host falha, mas volta a funcionar imediatamente. Quando $t = 60$, ele começa a reabrir as conexões de 0 a 4. Quando $t = 70$, ele reabre a conexão 5 utilizando o número de seqüência inicial 70, conforme solicitado. Durante os próximos 15 segundos, o host envia as TPDUs de dados de 70 a 80. Portanto, quando $t = 85$ uma nova TPDU com número de seqüência 80 e de conexão 5 entra na sub-rede. Infelizmente, a TPDU *X* ainda existe e, se ela chegar ao receptor antes da nova TPDU 80, a TPDU *X* será aceita e a TPDU 80 correta será rejeitada como uma duplicata.

Para evitar tais problemas, temos de impedir que os números de seqüência sejam utilizados (ou seja, atribuídos a novas TPDUs) durante um período de tempo *T*, antes de seu uso potencial como números de seqüência iniciais. As combinações inválidas de tempo e número de seqüência são mostradas como **região proibida**

na Figura 6.10(a). Antes de enviar uma TPDU em qualquer conexão, a entidade de transporte deve fazer a leitura do relógio e verificar se ele não está na região proibida.

O protocolo pode se complicar de duas maneiras. Se um host transmitir dados demais a uma velocidade muito alta em uma conexão recém-estabelecida, a verdadeira curva de número de seqüência *versus* tempo pode ter um crescimento mais acentuado que a curva de número de seqüência inicial *versus* tempo. Isso significa que a taxa máxima de transferência de dados em qualquer conexão é de uma TPDU para cada pulso do relógio. Isso também significa que a entidade de transporte deve esperar até que o relógio pulse para estabelecer uma nova conexão depois de uma retomada de funcionamento do host, a fim de evitar que o mesmo número seja utilizado duas vezes. Ambos os argumentos provam a importância do relógio pulsar a intervalos bem pequenos (alguns ms, ou menos).

Infelizmente, entrar na região proibida por ter enviado dados com muita rapidez não é a única maneira de arranjar problemas. A Figura 6.10(b) deixa claro que, para qualquer taxa de transmissão de dados menor que a velocidade do clock, a curva de números de seqüência reais utilizados *versus* tempo entrará eventualmente na região proibida a partir da esquerda. Quanto maior for a inclinação da curva do número de seqüência real, maior será o retardo do evento. Como já dissemos, antes de enviar cada TPDU, a entidade de transporte deve verificar se existe a possibilidade de entrar na região proibida e, nesse caso, retardar a TPDU por T segundos ou ressincronizar os números de seqüência.

O método baseado no relógio resolve o problema de duplicatas atrasadas para as TPDUs de dados; no entanto, para que ele seja realmente útil, é necessário que se estabeleça primeiro uma conexão. Como as TPDUs de controle também podem estar atrasadas, existe um problema potencial para se fazer com que ambos os lados concordem em relação ao número de seqüência inicial. Por exemplo,

suponha que as conexões sejam estabelecidas fazendo-se o host 1 enviar uma TPDU CONNECTION REQUEST contendo o número de seqüência inicial e o número da porta de destino propostos a um par remoto, o host 2. O receptor, o host 2, confirma essa solicitação enviando de volta uma TPDU CONNECTION ACCEPTED. Se a TPDU CONNECTION REQUEST se perder, mas uma duplicata atrasada de CONNECTION REQUEST chegar de repente ao host 2, a conexão será estabelecida de forma incorreta.

Para cuidar dessa questão, Tomlinson (1975) criou o **handshake de três vias**. Esse protocolo de estabelecimento não exige que ambos os lados comecem a enviar mensagens com o mesmo número de seqüência; portanto, ele pode ser usado com métodos de sincronização diferentes do método de relógio global. O procedimento de inicialização normal para o host 1 é ilustrado na Figura 6.11(a). O host 1 escolhe um número de seqüência inicial *x* e o envia em uma TPDU CONNECTION REQUEST para o host 2. Por sua vez, o host 2 responde com uma TPDU ACK que confirma *x* e anuncia seu próprio número de seqüência inicial, *y*. Por fim, o host 1 confirma o número de seqüência inicial escolhido pelo host 2 na primeira TPDU de dados que enviar.

Agora, vamos ver como o handshake de três vias funciona diante de duplicatas atrasadas de TPDUs de controle. Na Figura 6.11(b), a primeira TPDU é uma duplicata atrasada da primitiva CONNECTION REQUEST de uma antiga conexão. Essa TPDU chega ao host 2 sem o conhecimento do host 1. O host 2 reage a essa TPDU transmitindo uma TPDU ACK ao host 1, para verificar se o host 1 deseja realmente estabelecer uma nova conexão. Quando o host 1 rejeita a tentativa de conexão feita pelo o host 2, este percebe que foi enganado por uma duplicata atrasada e abandona a conexão. Dessa forma, uma duplicata atrasada não causa danos.

[arte: ver original p. 501]

[Dísticos]

[1]Host 1 Host 2

Tempo

CR (seq = x)

ACK (seq = y, ACK = x)

DATA (seq = x, ACK = y)

(a)

[2]Host 1 Host 2

Duplicata antiga

CR (seq = x)

ACK (seq = y, ACK = x)

REJECT (ACK = y)

(b)

[3]Host 1 Host 2

CR (seq = x)

Duplicata antiga

ACK (seq = y, ACK = x)

DATA (seq = x, ACK = z)

Duplicata antiga

REJECT (ACK = y)

(c)

[F]Figura 6.11

[FL] Três cenários de protocolos para o estabelecimento de uma conexão com o uso de um handshake de três vias. CR denota CONNECTION REQUEST. (a) Operação normal. (b) Duplicata antiga de CONNECTION REQUEST que surge repentinamente. (c) CONNECTION REQUEST e ACK duplicadas

A pior situação ocorre quando tanto uma CONNECTION REQUEST atrasada quanto uma ACK estão pairando na sub-rede. Esse caso é ilustrado na Figura 6.11(c). Como no exemplo anterior, o host 2 recebe uma CONNECTION REQUEST atrasada e responde a ela. É importante entender que o host 2 propôs o uso de *y* como número de seqüência inicial para o tráfego do host 2 ao host 1, o que implica que não existe qualquer TPDU que contenha o número de seqüência *y* ou que ainda existam confirmações para *y*. Quando a segunda TPDU atrasada chega ao host 2, o fato de *z* ter sido confirmado no lugar de *y* faz com que o host 2 também perceba que se trata de uma duplicata antiga. Portanto, não existe nenhuma combinação de TPDUs antigas que possa fazer o protocolo falhar e configurar uma conexão por acidente quando ela não for solicitada.

[T3] 6.2.3 Encerramento de conexões

Encerrar uma conexão é mais fácil do que estabelecê-la. No entanto, nesse procedimento há mais armadilhas do que se poderia esperar. Como já mencionamos, existem dois tipos de encerramento de conexão, o encerramento simétrico e o encerramento assimétrico. O encerramento assimétrico representa o funcionamento do sistema telefônico, ou seja, quando um dos interlocutores desliga, a conexão é interrompida. Em contraste, o encerramento simétrico trata a conexão como duas conexões unidirecionais isoladas e exige que cada uma seja encerrada separadamente.

O encerramento assimétrico é abrupto e pode resultar na perda de dados. Observe a Figura 6.12. Após o estabelecimento da conexão, o host 1 envia uma TPDU, que chega de forma correta ao host 2. Em seguida, o host 1 envia outra TPDU. Infelizmente, o host 2 gera uma primitiva DISCONNECT antes da segunda TPDU chegar, o que resulta no encerramento da conexão e na perda dos dados.

[arte: ver original p. 502]

[Dísticos]

[1]Host 1 Host 2

Tempo

CR

ACK

DATA

DATA

DR

Não há entrega de dados após uma solicitação de desconexão

[F]Figura 6.12

[FL] Desconexão abrupta com perda de dados

Fica clara a necessidade de se utilizar um protocolo de encerramento mais sofisticado para evitar a perda de dados. Uma forma de implementar essa estratégia é usar o encerramento simétrico, no qual cada direção da conexão é liberada de forma independente da outra. Nesse caso, um host pode continuar a receber dados mesmo depois de ter enviado uma TPDU DISCONNECT.

O encerramento simétrico é indicado quando cada processo tem uma quantidade fixa de dados a enviar e sabe com clareza quando terminou de enviá-los. Em outras situações, não é tão simples determinar que todo o trabalho foi concluído e que a conexão deve ser encerrada. É possível imaginar um protocolo no qual o host 1 diz: "Já terminei. Você também já terminou?" Se o host 2 responder: "Eu também já terminei. Adeus", a conexão poderá ser encerrada com segurança.

No entanto, nem sempre o protocolo funciona assim. Há um problema famoso que ilustra essa questão, chamado **problema dos dois exércitos**. Imagine que o exército branco está acampado em um vale, conforme ilustra a Figura 6.13. Acampados nas duas colinas ao redor do vale estão dois exércitos azuis. O

exército branco tem um contingente maior que cada exército azul, mas juntos os exércitos azuis são maiores que o exército branco. Se um dos exércitos azuis atacar sozinho, ele será derrotado pelo exército branco; porém, se atacarem o exército branco simultaneamente, os dois exércitos azuis serão vitoriosos.

[arte: ver original p. 503]

[Dísticos]

[1]A Exército azul nº 1

[2]A Exército azul nº 2

[3]B Exército branco

[F]Figura 6.13

[FL] O problema dos dois exércitos

Os exércitos azuis desejam sincronizar seus ataques. Entretanto, eles só podem se comunicar através de mensageiros que caminham pelo vale, onde podem ser capturados e a mensagem perdida (ou seja, eles têm de usar um canal de comunicação não confiável). A questão é: existe algum protocolo que permita aos exércitos azuis vencerem?

Suponhamos que o comandante do exército azul número 1 envie uma mensagem dizendo: "Proponho que nosso ataque seja ao amanhecer do dia 29 de março. Que tal?" Suponhamos também que a mensagem chegue a seu destino e que o comandante do exército azul número 2 concorde, e que sua mensagem chegue em segurança ao exército azul número 1. O ataque acontecerá? Provavelmente não, pois o comandante número 2 não tem como saber se o comandante número 1 recebeu sua resposta. Caso não tenha recebido a resposta, ele não vai atacar; portanto, seria inútil o exército azul número 2 começar a batalha sozinho.

Agora, vamos melhorar o protocolo tornando-o um handshake de três vias. O responsável pela proposta original deve confirmar a resposta fornecida. Supondo

que nenhuma mensagem se perca, o exército azul número 2 receberá a confirmação. No entanto, agora é a vez do comandante do exército número 1 hesitar. Afinal, ele não sabe se sua confirmação chegou ao destino e assim o exército azul número 2 não vai atacar. Poderíamos criar aqui o handshake de quatro vias, mas isso também não ajudaria.

Na verdade, podemos provar que não existe nenhum protocolo que funcione. No entanto, suponhamos que esse protocolo existisse. Nesse caso, deveríamos verificar se a última mensagem do protocolo é essencial ou não. Se a resposta for negativa, devemos removê-la (assim como todas as outras mensagens que não forem essenciais) até ficarmos com um protocolo no qual toda mensagem é essencial. O que acontecerá se a mensagem final não chegar a seu destino? Acabamos de afirmar que ela é essencial; portanto, se essa mensagem se perder. o ataque não ocorrerá. Como o transmissor não tem certeza de que a mensagem final foi recebida, ele não arrisca um ataque. Pior ainda, o outro exército azul sabe disso e também não ataca.

Para compreender a relevância do problema dos dois exércitos no encerramento de conexões, basta substituir "atacar" por "desconectar". Se nenhum dos lados estiver preparado para encerrar a conexão até estar convencido de que o outro lado também está pronto, o encerramento jamais acontecerá.

Na prática, é mais fácil correr riscos com o encerramento de conexões do que com um ataque ao exército branco; assim, a situação não é inteiramente insolúvel. A Figura 6.14 ilustra quatro cenários de encerramento utilizando o handshake de três vias. Ainda que não seja infalível, em geral esse protocolo é o mais adequado.

Na Figura 6.14(a), vemos uma situação normal em que um dos usuários envia uma TPDU DR (DISCONNECTION REQUEST) para dar início ao encerramento da conexão. Quando a TPDU chega, o receptor também retorna uma TPDU DR e

dispara um timer para o caso de a DR se perder. Quando essa DR chega, o transmissor original envia uma TPDU ACK e encerra a conexão. finalmente, quando a TPDU ACK chega, o receptor também encerra a conexão. Encerrar uma conexão significa que a entidade de transporte remove as informações sobre essa conexão de sua tabela de conexões atualmente abertas e envia algum tipo de sinal ao proprietário da conexão (o usuário de transporte). Essa ação é diferente de um usuário de transporte emitir uma primitiva DISCONNECT.

Se a última TPDU ACK for perdida, como ilustra a Figura 6.14(b), a situação poderá ser salva pelo timer. Quando o timer expirar, a conexão será encerrada de qualquer forma.

Agora, vamos considerar o caso em que a segunda DR se perde. O usuário que der início à desconexão não receberá a resposta esperada, sofrerá um timeout e terá de começar tudo outra vez. Na Figura 6.14(c), podemos ver como isso funciona, supondo que da segunda vez nenhuma TPDU se perdeu e que todas as TPDUs foram entregues de forma correta e em tempo.

Nosso último cenário, a Figura 6.14(d), é igual ao da Figura 6.14(c), exceto pelo fato de assumirmos agora que todas as tentativas repetidas de retransmitir a DR também falharam devido a TPDUs perdidas. Após *N* tentativas, o transmissor desiste e encerra a conexão. Nesse ínterim, o receptor sofre um timeout e também pára de funcionar.

Apesar de em geral ser suficiente, na teoria esse protocolo pode falhar, se a DR inicial e todas as *N* retransmissões se perderem. O transmissor desistirá e encerrará a conexão, enquanto o outro lado não saberá nada sobre todas as tentativas de desconexão e permanecerá ativo. Isso resulta em uma conexão semi-aberta.

[arte: ver original p. 505]

[Dísticos]

[1]Host 1 Host 2

Envia DR e inicializa timer DR Envia DR e inicializa timer

Encerra conexão DR

Envia ACK ACK Encerra conexão

(a)

[2]Host 1 Host 2

Envia DR e inicializa timer DR Envia DR e inicializa timer

Encerra conexão DR

Envia ACK ACK Perdida (Timeout) encerra conexão

(b)

[3]Host 1 Host 2

Envia DR e inicializa timer DR Envia DR e inicializa timer

Perdida DR

(Timeout) envia DR e inicializa timer DR Envia DR e inicializa timer

Encerra conexão DR

Envia ACK ACK Encerra conexão

(c)

[4]Host 1 Host 2

Envia DR e inicializa timer DR Envia DR e inicializa timer

Perdida

(Timeout) envia DR e inicializa timer Perdida

(N Timeouts) Encerra conexão

(Timeout) encerra conexão

(d)

[F]Figura 6.14

[FL] Quatro situações de protocolos para encerramento de uma conexão. (a) Caso normal de handshake de três vias. (b) ACK final perdida. (c) Resposta perdida. (d)

Resposta perdida e DRs subseqüentes perdidas

Poderíamos evitar esse problema não permitindo ao transmissor desistir após *N* tentativas e forçando-o a continuar tentando até obter resposta. Entretanto, se o outro lado tiver permissão para entrar em timeout, o transmissor continuará tentando para sempre, pois nenhuma resposta será recebida. Se não permitirmos que o lado receptor entre em timeout, o protocolo se comportará como na Figura 6.14(b).

Uma forma de extinguir conexões semi-abertas é criar uma regra informando que, se nenhuma TPDU chegar durante um determinado número de segundos, a conexão será encerrada automaticamente. Dessa forma, se um lado encerrar a conexão, o outro perceberá a falta de atividade e também encerrará a conexão. É óbvio que, se essa regra for utilizada, será necessário que cada entidade de transporte tenha um timer que será interrompido e depois reiniciado sempre que uma TPDU for enviada. Se esse timer expirar, uma TPDU fictícia será transmitida para evitar que o outro lado se desconecte. Por outro lado, se a regra de encerramento automática for usada e muitas TPDUs fictícias seguidas se perderem em uma conexão inativa, a transmissão será encerrada automaticamente, um lado de cada vez.

Não trataremos mais desse assunto, mas já deve estar claro que encerrar uma conexão sem perda de dados não é tão simples quanto parecia à primeira vista.

[T3] 6.2.4 Controle de fluxo e uso de buffers

Depois de examinar o estabelecimento e o encerramento de conexões, vamos ver como elas são administradas enquanto estão ativas. Uma das questões fundamentais já surgiu antes: o controle de fluxo. Sob alguns aspectos, o problema do controle de fluxo na camada de transporte é igual ao da camada de

enlace de dados mas, em outros aspectos, ele é diferente. A semelhança básica é que em ambas as camadas é necessário utilizar uma janela deslizante ou outro esquema qualquer em cada conexão para impedir que um transmissor rápido sobrecarregue um receptor lento. A principal diferença é que, em geral, um roteador tem relativamente poucas linhas, enquanto um host pode ter inúmeras conexões. É essa diferença que impossibilita a implementação do uso de buffers na camada de transporte, como ocorre na camada de enlace de dados.

Nos protocolos de enlace de dados do Capítulo 3, os quadros eram armazenados em buffers, tanto no roteador transmissor quanto no receptor. Por exemplo, no protocolo 6 tanto o transmissor quando o receptor têm de dedicar *MaxSeq* + 1 buffers a cada linha, metade para a entrada de dados e metade para a saída. Para um host com um máximo de, digamos, 64 conexões e um número de seqüência de 4 bits, esse protocolo necessitaria de 1024 buffers.

Na camada de enlace de dados, o lado transmissor deve armazenar os quadros de saída em buffers, pois talvez seja necessário retransmiti-los. Se a sub-rede oferecer serviço de datagrama, a entidade de transporte transmissora também deverá ter buffers pelo mesmo motivo. Se souber que o transmissor mantém buffers para todas as TPDUs até que elas sejam confirmadas, o receptor poderá ou não dedicar determinados buffers a conexões específicas, de acordo com sua conveniência. Por exemplo, o receptor pode manter um único pool de buffers a ser compartilhado por todas as conexões. Quando uma TPDU chega, há uma tentativa de adquirir um novo buffer dinamicamente. Se houver um buffer disponível, a TPDU será aceita; caso contrário, ela será descartada. Como o transmissor está preparado para retransmitir TPDUs perdidas pela sub-rede, não há danos no fato de o receptor abandonar as TPDUs, mesmo que alguns recursos sejam desperdiçados. O transmissor continua tentando até conseguir uma confirmação.

Em suma, se o serviço de rede não for confiável, o transmissor deverá ter buffers para todas as TPDUs enviadas, exatamente como acontece na camada de enlace de dados. Entretanto, com um serviço de rede confiável, há outras opções. Em particular, se o transmissor souber que o receptor sempre tem espaço em buffers, ele não precisará manter cópias das TPDUs enviadas. No entanto, se o receptor não puder garantir que todas as TPDUs recebidas serão aceitas, o transmissor deverá utilizar buffers. Nesse último caso, o transmissor não poderá confiar na confirmação da camada de rede, pois essa confirmação significa apenas que a TPDU chegou ao destino, e não que ela tenha sido aceita. Voltaremos a esse importante ponto mais adiante.

Mesmo que o receptor tenha concordado em utilizar buffers, ainda existe a questão do tamanho dos buffers. Se a maioria das TPDUs tiver aproximadamente o mesmo tamanho, é natural organizar os buffers em um grupo de buffers de mesmo tamanho, com uma TPDU para cada um, como mostra a Figura 6.15(a). Entretanto, se houver uma grande variação no tamanho das TPDUs, desde alguns caracteres digitados em um terminal até milhares de caracteres obtidos em transferências de arquivos, um pool de buffers de tamanho fixo apresentará problemas. Se o tamanho do buffer for escolhido a partir do tamanho da maior TPDU possível, haverá desperdício de espaço sempre que uma TPDU pequena for recebida. Se o tamanho de buffer escolhido for menor que o tamanho máximo da TPDU, vários buffers serão necessários para as TPDUs mais longas, havendo uma conseqüente complexidade no gerenciamento.

[arte: ver original p. 507]

[Dísticos]

[1] (a) (b)

[2] Espaço não utilizado

[3] TPDU 1

TPDU 2

TPDU 3

TPDU 4

(c)

[F]Figura 6.15

[FL] (a) Buffers de tamanho fixo encadeados. (b) Buffers de tamanho variável encadeados. (c) Um grande buffer circular por conexão

Outra abordagem para o problema do tamanho dos buffers é usar buffers de tamanho variável, como mostra a Figura 6.15(b). A vantagem é a melhor utilização da memória, à custa de um gerenciamento de buffers mais complicado. Uma terceira possibilidade é dedicar um grande buffer circular a cada conexão, como é ilustra a Figura 6.15(c). Esse sistema também faz um bom uso da memória quando todas as conexões têm uma carga muito pesada, mas se mostra ineficaz quando há algumas conexões com pouca carga.

A opção ideal entre os buffers de origem e os buffers de destino depende do tipo de tráfego transportado pela conexão. Para um tráfego em rajadas de baixa largura de banda, como o produzido por um terminal interativo, a melhor opção não é dedicar buffers, mas sim adquiri-los dinamicamente em ambas as extremidades. Como o transmissor não pode ter certeza de que o receptor será capaz de adquirir um buffer, o transmissor deve reter uma cópia da TPDU até que ela seja confirmada. Por outro lado, para a transferência de arquivos e outros tipos de tráfego de alta largura de banda, a melhor opção é o receptor dedicar uma janela de buffers inteira, permitindo que o fluxo de dados se dê na velocidade máxima. Portanto, para um tráfego em rajadas de baixa largura de banda, é melhor que os buffers estejam no transmissor, enquanto para um tráfego suave de alta largura de banda, a melhor opção é que os buffers estejam

no receptor.

À medida que as conexões são abertas e fechadas e que o padrão de tráfego se altera, o transmissor e o receptor precisam ajustar dinamicamente suas alocações de buffers. Conseqüentemente, o protocolo de transporte deve permitir que o host transmissor solicite espaço em buffer na outra extremidade da conexão. Os buffers poderiam ser alocados por conexão ou coletivamente, para todas as conexões em execução entre os dois hosts. Em contrapartida, o receptor, conhecendo a situação de seus buffers (mas sem conhecer o tráfego oferecido) poderia dizer ao transmissor: "Tenho *X* buffers reservados para você". Se o número de conexões abertas aumentar, talvez seja necessário reduzir uma alocação de buffer; portanto, o protocolo deve oferecer essa possibilidade.

Um modo razoavelmente genérico de gerenciar a alocação dinâmica de buffers é desvincular o gerenciamento dos buffers das confirmações, ao contrário do que ocorre com os protocolos de janela deslizante do Capítulo 3. Na verdade, o gerenciamento dinâmico de buffers significa usar uma janela de tamanho variável. Inicialmente, o transmissor solicita um determinado número de buffers, com base em suas necessidades. Em seguida, de acordo com o número solicitado, o receptor oferece todos os buffers de que dispõe. Sempre que enviar uma TPDU, o transmissor deverá decrementar sua alocação, parando por completo quando a alocação chegar a zero. Em seguida, o receptor transmite (por piggyback) as confirmações e as alocações de buffers no tráfego reverso.

A Figura 6.16 mostra um exemplo de como o gerenciamento dinâmico de janelas poderia funcionar em uma sub-rede de datagramas com números de seqüência de 4 bits. Vamos supor que as informações de alocação de buffers viajem em TPDUs separadas, como mostramos, e que elas não sejam transportadas com o tráfego reverso. Inicialmente, *A* quer oito buffers, dos quais só recebe quatro. Em seguida, *A* envia três TPDUs, sendo que a terceira é perdida. A TPDU 6 confirma a

recepção de todas as TPDUs até o número de seqüência 1 inclusive, permitindo que *A* libere esses buffers, e informa a *A* que ele tem permissão para enviar mais três TPDUs, começando pela TPDU seguinte à de número 1 (isto é, as TPDUs 2, 3 e 4). *A* sabe que já enviou a TPDU de número 2 e assim imagina que pode enviar as TPDUs 3 e 4, o que acaba por fazer. Nesse ponto, ele é bloqueado e deve aguardar a alocação de mais buffers. Entretanto, pode haver retransmissões (linha 9) induzidas por timeouts durante o bloqueio, pois elas utilizam buffers que já haviam sido alocados. Na linha 10, *B* confirma a recepção de todas as TPDUs até 4 inclusive, mas se recusa a permitir que *A* continue. Essa situação é impossível com os protocolos de janela fixa do Capítulo 3. A próxima TPDU de *B* para *A* aloca outro buffer e permite que *A* continue.

Podem surgir problemas potenciais com esquemas de alocação de buffers desse tipo em redes de datagramas, caso ocorra a perda de TPDUs de controle. Observe a linha 16. *B* já alocou mais buffers para *A*, mas a TPDU da alocação foi perdida. Como as TPDUs de controle não seguem uma seqüência nem sofrem timeout, *A* está em um impasse. Para evitar que isso aconteça, cada host deve enviar periodicamente TPDUs de controle informando o status dos buffers e das confirmações em cada conexão. Dessa forma, o impasse será resolvido mais cedo ou mais tarde.

Até agora, partimos da suposição tácita de que o único limite imposto sobre a taxa de dados do transmissor é o espaço em buffer disponível no receptor. Se os preços dos chips de memória continuarem a cair drasticamente, talvez seja viável equipar os hosts com tanta memória, que a falta de buffers passará a ser um problema raro, isso se não desaparecer totalmente.

[arte: ver original p. 509]

[Dísticos]

[1] A Mensagem

1 → <request 8 buffers>

2 ← <ack = 15, buf = 4>

3 → <seq = 0, data = m0>

4 → <seq = 1, data = m1>

5 → <seq = 2, data = m2>

6 ← <ack = 1, buf = 3>

7 → <seq = 3, data = m3>

8 → <seq = 4, data = m4>

9 → <seq = 2, data = m2>

10 ← <ack = 4, buf = 0>

11 ← <ack = 4, buf = 1>

12 ← <ack = 4, buf = 2>

13 → <seq = 5, data = m5>

14 → <seq = 6, data = m6>

15 ← <ack = 6, buf = 0>

16 ... <ack = 6, buf = 4>

<u>B</u> <u>Comentários</u>

→ A quer 8 buffers

← B concede apenas as mensagens 0 a 3

→ A tem agora 3 buffers sobrando

→ A tem agora 2 buffers sobrando

... Mensagem perdida, mas A pensa que tem 1 sobrando

← B confirma 0 e 1, permite 2 a 4

→ A tem 1 buffer sobrando

→ A tem 0 buffers sobrando e tem de parar

→ A chega ao tempo limite (timeout) e retransmite

← Tudo confirmado, mas A ainda está bloqueado

← Agora A pode enviar 5

← B encontrou um novo buffer em outro lugar

→ A tem 1 buffer sobrando

→ Agora A está bloqueado outra vez

← A ainda está bloqueado

← Possível impasse

[F]Figura 6.16

[FL] Alocação dinâmica de buffers. As setas mostram o sentido da transmissão. As reticências (...) indicam a perda de uma TPDU

Quando o espaço em buffer deixar de limitar o fluxo máximo, surgirá outro gargalo: a capacidade de transporte da sub-rede. Se os roteadores adjacentes puderem trocar dados em uma velocidade de no máximo x pacotes/segundo, e se houver k caminhos disjuntos entre um par de hosts, esses hosts não poderão trocar mais de kx TPDUs/segundo, independente do espaço em buffer disponível em cada extremidade da conexão. Se o transmissor forçar muito a transmissão (ou seja, enviar mais de kx TPDUs/segundo), a sub-rede ficará congestionada, pois não será capaz de entregar as TPDUs na mesma velocidade em que elas chegam.

É necessário um mecanismo baseado na capacidade de transporte da sub-rede, em vez do mecanismo baseado na capacidade dos buffers do receptor. O mecanismo de controle de fluxo deve, obviamente, ser usado no transmissor para evitar que haja muitas TPDUs não confirmadas pendentes ao mesmo tempo. Belsnes (1975) propôs o uso de um esquema de controle de fluxo com uma janela deslizante, no qual o transmissor ajusta dinamicamente o tamanho da janela à capacidade de transporte da rede. Se a rede puder administrar c TPDUs/segundo e o tempo de ciclo (incluindo transmissão, propagação, enfileiramento,

processamento no receptor e retorno da confirmação) for *r*, a janela do transmissor deverá ser *cr*. Com uma janela desse tamanho, o transmissor opera com toda a capacidade do pipeline. Qualquer pequeno decréscimo no desempenho da rede bloqueará o fluxo.

Para ajustar o tamanho da janela periodicamente, o transmissor pode monitorar os dois parâmetros e calcular o tamanho de janela desejado. É possível determinar a capacidade de transporte simplesmente contando-se o número de TPDUs confirmadas durante um intervalo de tempo e dividindo-se esse valor pelo intervalo de tempo. Durante a medição, o transmissor deverá enviar as TPDUs o mais rápido que puder, de modo que a capacidade de transporte da rede, e não a baixa taxa de entrada, seja o fator limitador da taxa de confirmação. É possível medir com precisão o tempo necessário para que uma TPDU retransmitida seja confirmada, mantendo-se uma média atualizada. Como a capacidade da rede disponível para qualquer fluxo dado varia com o tempo, o tamanho da janela deve ser ajustado com freqüência para acompanhar as mudanças na capacidade de transporte. Como veremos mais adiante, a Internet utiliza um esquema semelhante.

[T3] 6.2.5 Multiplexação

A multiplexação de várias conversações em conexões, circuitos virtuais e enlaces físicos tem um papel importante em diversas camadas da arquitetura de rede. Na camada de transporte, a necessidade da multiplexação pode surgir de diversas formas. Por exemplo, se houver apenas um endereço de rede disponível em um host, todas as conexões de transporte nessa máquina terão de utilizá-lo. Ao chegar uma TPDU, é necessário encontrar algum meio de descobrir a qual processo ela deve ser entregue. Essa situação, denominada **multiplexação ascendente**, é ilustrada na Figura 6.17(a). Nessa figura, quatro conexões de

transporte distintas utilizam a mesma conexão de rede (por exemplo, um endereço IP) para o host remoto.

[arte: ver original p. 510]

[Dísticos]

[1]Camada

Linhas de roteadores

Para o roteador

(a)

[2]Endereço de transporte

Endereço de rede

(b)

[F]Figura 6.17

[FL] (a) Multiplexação ascendente. (b) Multiplexação descendente

A multiplexação também pode ser útil na camada de transporte por outra razão. Por exemplo, suponha que uma sub-rede utilize circuitos virtuais internamente e imponha uma taxa máxima de dados sobre cada um. Se um usuário necessitar de maior largura de banda do que um único circuito virtual pode fornecer, uma saída será abrir várias conexões de rede e distribuir o tráfego entre elas em rodízio, como indica a Figura 6.17(b). Esse modo de operação é chamado **multiplexação descendente**. Com k conexões de rede abertas, a largura de banda efetiva é aumentada k vezes. Um exemplo comum de multiplexação descendente ocorre com alguns usuários domésticos que têm linhas ISDN. Essa linha permite a utilização de duas conexões separadas de 64 kbps cada uma. Empregando-se ambas para se conectar a um provedor de serviços da Internet e dividindo-se o tráfego entre as duas linhas, pode-se alcançar uma largura de banda efetiva de 128 kbps.

[T3] 6.2.6 Recuperação de desastres

Se os hosts e roteadores estiverem sujeitos a interrupções em seu funcionamento, a recuperação dessas panes se tornará uma questão muito importante. Se a entidade de transporte estiver inteiramente contida nos hosts, a recuperação depois de falhas na rede ou no roteador será simples. Se a camada de rede oferecer serviço de datagramas, as entidades de transporte estarão sempre esperando por TPDUs perdidas e saberão como lidar com elas. Se a camada de rede oferecer um serviço orientado a conexões, a perda de um circuito virtual será contornada estabelecendo-se um novo circuito e perguntando-se à entidade de transporte remota quais TPDUs ela recebeu e quais não recebeu. As TPDUs não recebidas poderão ser retransmitidas.

Um problema mais complexo é como recuperar o funcionamento depois de uma pane no host. Em particular, talvez seja preferível que os clientes possam continuar funcionando quando os servidores falharem e forem rapidamente reinicializados em seguida. Para ilustrar a dificuldade, vamos supor que um host, o cliente, esteja enviando um arquivo muito grande a outro host, o servidor de arquivos, utilizando um simples protocolo stop-and-wait. A camada de transporte do servidor simplesmente passa as TPDUs que chegam para usuário de transporte, uma a uma. No meio da transmissão, o servidor sai do ar. Quando ele volta a funcionar, suas tabelas são reinicializadas; desse modo, ele não sabe mais como identificar com precisão onde parou.

Na tentativa de recuperar seu estado anterior, o servidor pode transmitir uma TPDU de difusão a todos os outros hosts, comunicando seu problema e solicitando que seus clientes o informem sobre o status de todas as conexões abertas. Cada cliente pode estar em uma das seguintes situações: uma TPDU pendente, *S*1, ou nenhuma TPDU pendente, *S*0. Com base apenas nessas

informações de estado, o cliente tem de decidir se deve ou não retransmitir a TPDU mais recente.

À primeira vista parece óbvio que o cliente só deve retransmitir se houver uma TPDU não confirmada pendente (isto é, se ele se encontrar no estado *S*1) durante a queda do servidor. contudo, uma análise mais detalhada revela as dificuldades dessa abordagem simplista. Por exemplo, considere a situação na qual a entidade de transporte do servidor primeiro envia uma confirmação e depois, quando a confirmação tiver sido enviada, efetua uma gravação no processo de aplicação. Gravar uma TPDU no fluxo de saída e enviar uma confirmação são dois eventos distintos e indivisíveis que não podem ser realizados simultaneamente. Se ocorrer uma queda após o envio da confirmação, mas antes da gravação ser feita, o cliente receberá a confirmação e assim ficará no estado *S*0 quando chegar o anúncio de que o funcionamento foi recuperado. Conseqüentemente, o cliente não retransmitirá, porque imagina (incorretamente) que a TPDU chegou. Essa decisão do cliente leva à perda de uma TPDU.

A essa altura, você deve estar pensando: "É fácil resolver esse problema. Basta reprogramar a entidade de transporte para gravar primeiro a TPDU e depois enviar a confirmação". Tente outra vez. Imagine que a gravação foi feita mas a falha do servidor ocorreu antes de ser possível enviar a confirmação. O cliente estará no estado *S*1 e portanto retransmitirá, acarretando uma duplicata da TPDU não detectada no fluxo de saída para o processo de aplicação do servidor.

Independente da forma como o cliente e o servidor são programados, sempre haverá situações em que o protocolo não recuperará o funcionamento de forma apropriada. O servidor poderá ser programado de duas maneiras: para confirmar primeiro ou para gravar primeiro. O cliente poderá ser programado de quatro formas: para sempre retransmitir a última TPDU, para nunca retransmitir a última TPDU, para retransmitir somente no estado *S*0 ou para retransmitir somente no

estado *S*1. Isso nos dá oito combinações mas, como veremos, para cada combinação existe algum conjunto de eventos que faz o protocolo falhar.

Há três eventos possíveis no servidor: enviar uma confirmação (*A*), gravar no um processo de saída (*W*) e sofrer uma pane (*C*). Os três eventos podem ocorrer em seis ordens distintas: *AC*(*W*), *AWC*, *C*(*AW*), *C*(*WA*), *WAC* e *WC*(*A*); os parênteses são usados para indicar que nem *A* nem *W* podem seguir *C* (ou seja, não há qualquer evento após uma pane). A Figura 6.18 mostra todas as oito combinações de estratégias do cliente e do servidor, e as seqüências de eventos válidas para cada uma delas. Observe que, para cada estratégia, existe alguma seqüência de eventos que leva à falha do protocolo. Por exemplo, se o cliente sempre retransmitir, o evento *AWC* gerará uma duplicata não detectada, mesmo que os dois outros eventos funcionem perfeitamente.

Tornar o protocolo mais elaborado também não ajuda muito. Ainda que o cliente e o servidor troquem várias TPDUs antes de o servidor tentar gravar, de forma que o cliente saiba exatamente o que está para acontecer, o cliente não terá meios para saber se ocorreu uma pane imediatamente antes ou imediatamente após a gravação. A conclusão é inevitável: sob nossas regras básicas de não haver eventos simultâneos, a queda e a recuperação do host não podem ser realizadas de forma transparente para as camadas mais altas.

Em termos mais genéricos, esse resultado pode ser reafirmado como o fato de que a recuperação de uma queda da camada *N* só pode ser feita pela camada $N + 1$, e mesmo assim somente se a camada mais elevada mantiver um volume suficiente de informações sobre o status. Como mencionamos anteriormente, a camada de transporte pode se recuperar de falhas na camada de rede, desde que cada extremidade da conexão tenha uma idéia do ponto em que está.

[arte: ver original p. 513]

[Dísticos]

[1]Estratégia usada pelo host receptor

Primeiro confirma, depois grava Primeiro grava, depois confirma

[2]Estratégia usada pelo host transmissor

Sempre retransmitir

Nunca retransmitir

Retransmitir em S0

Retransmitir em S1

[3]

AC(W) AWC C(AW)

OK DUP OK

LOST OK LOST

OK DUP LOST

LOST OK OK

[4]

C(WA) WAC WC(A)

OK DUP DUP

LOST OK OK

LOST DUP OK

OK OK DUP

[5]OK = Protocolo funciona corretamente

DUP = Protocolo gera uma mensagem duplicada

LOST = Protocolo perde uma mensagem

[F]Figura 6.18

[FL] Diferentes combinações de estratégias do cliente e do servidor

Esse problema nos leva à questão do que significa de fato a chamada confirmação fim a fim. Em princípio, o protocolo de transporte é fim a fim, pois não é

encadeado como as camadas inferiores. Considere agora o caso de um usuário que solicita transações relativas a um banco de dados remoto. Suponha que a entidade de transporte remota esteja programada de modo a passar primeiro as TPDUs para a camada imediatamente superior e só então enviar a confirmação. Até mesmo nesse caso, o fato de uma confirmação ter sido recebida na máquina do usuário não quer dizer necessariamente que o host remoto funcionou por tempo suficiente para atualizar o banco de dados. Uma confirmação fim a fim verdadeira, cujo recebimento indica que o trabalho foi realmente realizado e cuja falta indica que ele não foi cumprido, talvez seja algo impossível de se alcançar. Esse assunto é discutido com mais detalhes por Saltzer *et al.* (1984).

[T2] 6.3 Um protocolo de transporte simples

Para tornar mais concretas as idéias que discutimos até agora, vamos estudar em detalhes nesta seção um exemplo da camada de transporte. As primitivas de serviço abstratas que utilizaremos são as primitivas orientadas a conexões da Figura 6.2. A escolha dessas primitivas orientadas a conexões torna o exemplo semelhante ao do conhecido protocolo TCP (embora seja mais simples que este último).

[T3] 6.3.1 Exemplo de primitivas de serviço

Nosso primeiro problema é expressar essas primitivas de transporte de forma concreta. A primitiva CONNECT é fácil: teremos um procedimento de biblioteca *connect* que pode ser chamado com os parâmetros necessários para estabelecer uma conexão. Os parâmetros são os TSAPs locais e remotos. Durante o processo, o responsável pela chamada é bloqueado (ou seja, é suspenso), enquanto a entidade de transporte tenta estabelecer a conexão. Se a conexão for bem-sucedida, o responsável pela chamada será desbloqueado e poderá dar início à

transmissão de dados.

Quando um processo deseja estar apto a receber chamadas, ele chama *listen* especificando um determinado TSAP a ser escutado. Com isso, o processo fica bloqueado até que algum processo remoto tente estabelecer uma conexão com seu TSAP.

Observe que esse modelo é altamente assimétrico. Um lado é passivo, executando um procedimento *listen* e aguardando até que algo aconteça. O outro lado é ativo e dá início à conexão. Uma questão interessante é o que fazer se o lado ativo iniciar os procedimentos. Uma estratégia é fazer a tentativa de conexão falhar se não houver alguém na escuta no TSAP remoto. Outra estratégia é manter o bloco iniciador (talvez para sempre) até que haja alguém escutando.

Um meio-termo usado no nosso exemplo é manter a solicitação de conexão na extremidade receptora durante um determinado intervalo de tempo. Se um processo desse host chamar um procedimento *listen* antes do timer expirar, a conexão será estabelecida; caso contrário, ela será rejeitada, o responsável pela chamada será desbloqueado e receberá uma mensagem de erro.

Para encerrar uma conexão, utilizaremos um procedimento *disconnect*. Quando os dois lados se desconectarem, a conexão será encerrada. Em outras palavras, usaremos um modelo de encerramento simétrico.

A transmissão de dados tem exatamente o mesmo problema do estabelecimento da conexão, ou seja, a transmissão é ativa mas a recepção é passiva. Utilizaremos a mesma solução que usamos no estabelecimento da conexão: uma chamada ativa *send*, que transmite dados, e uma chamada passiva *receive* que permanece bloqueada até a chegada de uma TPDU.

Portanto, nossa definição concreta de serviço consiste em cinco primitivas: CONNECT, LISTEN, DISCONNECT, SEND e RECEIVE. Cada primitiva corresponde exatamente a um procedimento de biblioteca que executa a primitiva. Os

parâmetros das primitivas de serviço e dos procedimentos de biblioteca são os seguintes:

[TD]

numcon = LISTEN (local)

numcon = CONNECT (local, remoto)

status = SEND (numcon, buffer, bytes)

status = RECEIVE (numcon, buffer, bytes)

status = DISCONNECT (numcon) [TN]

A primitiva LISTEN anuncia a disposição do chamador para aceitar solicitações de conexão dirigidas ao TSAP indicado. O usuário da primitiva permanece bloqueado até que haja uma tentativa de conexão com ele. Não há timeout.

A primitiva CONNECT utiliza dois parâmetros, um TSAP local (isto é, um endereço de transporte) denominado *local*, e um TSAP remoto, chamado *remoto*, e tenta estabelecer uma conexão de transporte entre os dois. Se a tentativa for bem-sucedida, ela retornará em *numcom* um número não negativo usado para identificar a conexão em chamadas subseqüentes. Se a tentativa falhar, a causa da falha será colocada em *numcom* como um número negativo. Em nosso modelo simples, cada TSAP só pode participar de uma conexão de transporte; portanto, uma possível razão para a falha é um dos endereços de transporte estar em uso no momento. Outras razões podem ser: o host remoto está fora do ar, o endereço local é inválido ou o endereço remoto é inválido.

A primitiva SEND transmite o conteúdo do buffer como uma mensagem sobre a conexão de transporte indicada, em várias unidades, se necessário. Possíveis erros retornados em *status* são falta de conexão, endereço de buffer inválido ou contagem negativa.

A primitiva RECEIVE indica o desejo do chamador de receber dados. O tamanho da mensagem da entrada é expresso em *bytes*. Se o processo remoto tiver encerrado

a conexão ou o endereço do buffer for inválido (por exemplo, se ele estiver fora do programa do usuário), *status* mostrará um código de erro indicando a natureza do problema.

A primitiva DISCONNECT encerra uma conexão de transporte. O parâmetro *numcon* especifica qual delas. Os erros possíveis são: *numcon* pertence a outro processo ou *numcon* não é um identificador de conexão válido. O código de erro, ou 0 para indicar uma desconexão bem-sucedida, é retornado em *status*.

[T3] 6.3.2 Exemplo de entidade de transporte

Antes de verificar o código da entidade de transporte do exemplo, certifique-se de ter compreendido que esse exemplo é análogo aos exemplos apresentados anteriormente no Capítulo 3, pois sua finalidade é mais pedagógica do que uma proposta séria. Muitos dos detalhes técnicos (tais como uma ampla verificação de erros) que seriam necessários em um sistema real foram omitidos para tornar o processo mais simples.

A camada de transporte utiliza as primitivas do serviço de rede para enviar e receber TPDUs. Nesse exemplo, precisamos escolher quais serão as primitivas do serviço de rede a serem usadas. Uma opção poderia ser o serviço de datagramas não confiável. Não o escolhemos para simplificar o exemplo. Com o serviço de datagramas não confiável, o código de transporte se tornaria longo e complexo, lidando principalmente com pacotes perdidos e atrasados. Além disso, a maior parte dessas idéias já foi discutida em detalhes no Capítulo 3.

Em vez disso, optamos por utilizar um serviço de rede confiável e orientado a conexões. Dessa forma, podemos nos concentrar nos aspectos do transporte que não ocorrem nas camadas mais baixas. Isso inclui o estabelecimento da conexão, o encerramento da conexão e o gerenciamento de crédito, entre outros aspectos. Um serviço de transporte simples instalado sobre uma rede ATM teria uma

concepção semelhante a essa.

Em geral, a entidade de transporte pode fazer parte do sistema operacional do host ou pode ser um pacote de rotinas de biblioteca executadas dentro do espaço de endereços do usuário. Para simplificar, nosso exemplo foi programado como se fosse um pacote de biblioteca, mas as alterações necessárias para torná-lo parte do sistema operacional são mínimas (principalmente na maneira como os buffers dos usuários são acessados).

No entanto, vale a pena notar que nesse exemplo a "entidade de transporte" na verdade não é uma entidade distinta, mas faz parte do processo do usuário. Em particular, quando o usuário executa uma primitiva que se bloqueia, como por exemplo LISTEN, a entidade de transporte também é bloqueada. Esse projeto é adequado a um host com apenas um processo de um único usuário. Entretanto, para um host com vários usuários, seria mais natural tornar a entidade de transporte um processo separado, distinto de todos os processos do usuário.

A interface para a camada de rede é estabelecida por meio de procedimentos *to_net* e *from_net* (não mostrados). Cada um tem seis parâmetros. O primeiro é o identificador da conexão, que faz o mapeamento de um para um entre as conexões e os circuitos virtuais da rede. Depois vêm os bits *Q* e *M* que, quando definidos como 1, indicam respectivamente uma mensagem de controle, e que mais dados dessa mensagem virão em seguida no próximo pacote. O tipo de pacote é o parâmetro seguinte; ele é escolhido dentre os seis tipos apresentados na Figura 6.19. Por último, temos um ponteiro para os dados propriamente ditos e um valor inteiro que indica o número de bytes de dados.

[arte: ver original p. 516]

[T]Tabela

Pacote de rede	**Significado**
CALL REQUEST	Enviado para estabelecer uma conexão

CALL ACCEPTED Resposta a CALL REQUEST

CLEAR REQUEST Enviado para encerrar uma conexão

CLEAR CONFIRMATION Resposta a CLEAR REQUEST

DATA Usado para transportar dados

CREDIT Pacote de controle para gerenciamento da janela

[F]Figura 6.19

[FL] Os pacotes da camada de rede usados no nosso exemplo

Nas chamadas a *to_net*, a entidade de transporte fornece todos os parâmetros que deverão ser lidos pela camada de rede; já nas chamadas a *from_net*, a camada de rede desmembra um pacote recebido para a entidade de transporte. Ao passar as informações como parâmetros de procedimentos, em vez de enviar o próprio pacote de entrada ou saída, a camada de transporte se protege contra os detalhes do protocolo da camada de rede. Se a entidade de transporte tentar enviar um pacote quando a janela deslizante do circuito virtual estiver cheia, o pacote ficará suspenso em *to_net* até que haja espaço na janela. Esse mecanismo é transparente para a entidade de transporte e é controlado pela camada de rede através de comandos análogos aos comandos *enable_transport_layer* e *disable_transport_layer* descritos nos protocolos do Capítulo 3. O gerenciamento da janela da camada de pacotes também é feito pela camada de rede.

Além desse mecanismo transparente de suspensão, existem ainda os procedimentos explícitos *sleep* e *wakeup* (não mostrados), que são chamados pela entidade de transporte. O procedimento *sleep* é chamado quando a entidade de transporte permanece logicamente bloqueada, aguardando o acontecimento de um evento externo, em geral a chegada de um pacote. Depois que *sleep* é chamado, a entidade de transporte (e, é claro, o processo do usuário) interrompe sua execução.

O código real da entidade de transporte é mostrado na Figura 6.20. Cada conexão sempre se encontra em um dos seguintes estados:

1. IDLE — A conexão ainda não foi estabelecida.
2. WAITING — A primitiva CONNECT foi executada e um pacote CALL REQUEST foi enviado.
3. QUEUED — Um pacote CALL REQUEST chegou, mas a primitiva LISTEN ainda não foi executada.
4. ESTABLISHED — A conexão foi estabelecida.
5. SENDING — O usuário está aguardando permissão para enviar um pacote.
6. RECEIVING — Foi executada uma primitiva RECEIVE.
7. DISCONNECTING — Foi executada uma primitiva DISCONNECT local.

Podem surgir transições entre estados quando ocorrer um dos seguintes eventos: a execução de uma primitiva, a chegada de um pacote ou a expiração do timer.

Os procedimentos mostrados na Figura 6.20 são de dois tipos. A maioria deles pode ser chamada diretamente por programas do usuário. Porém, *packet_arrival* e *clock* são diferentes. Eles são acionados de forma espontânea por eventos externos: a chegada de um pacote e o pulsar do clock, respectivamente. Na prática, eles são rotinas de interrupção. Vamos supor que eles nunca são chamados durante a execução de um procedimento da entidade de transporte. Apenas quando o processo do usuário está suspenso ou sendo executando fora da entidade de transporte, essas rotinas podem ser invocadas. Essa propriedade é crucial para o funcionamento correto do código.

A existência do bit *Q* (Qualificador) no cabeçalho do pacote permite evitar o overhead de um cabeçalho de protocolo de transporte. As mensagens de dados comuns são enviadas como pacotes de dados com $Q = 0$. As mensagens de controle do protocolo de transporte, das quais só existe uma no nosso exemplo (CREDIT), são enviadas como pacotes de dados com $Q = 1$. Essas mensagens de

controle são detectadas e processadas pela entidade de transporte receptora.

A principal estrutura de dados utilizada pela entidade de transporte é o array *conn*, que dispõe de um registro para cada conexão potencial. O registro mantém o estado da conexão, incluindo os endereços de transporte em cada extremidade da conexão, o número de mensagens enviadas e recebidas na conexão, o estado atual, o ponteiro para o buffer do usuário, o número de bytes das mensagens atuais enviadas ou recebidas até agora, um bit indicando que o usuário remoto executou uma primitiva DISCONNECT, um timer, e ainda um contador de permissões usado para habilitar o envio de mensagens. Nem todos esses campos são empregados no nosso exemplo simples, mas uma entidade de transporte completa necessitaria de todos eles, e talvez de mais alguns. supõem-se que cada entrada *conn* é inicializada com o estado *IDLE*.

Quando o usuário chama uma primitiva CONNECT, a camada de rede é instruída a enviar um pacote CALL REQUEST até a máquina remota, e o usuário permanece suspenso. Quando o pacote CALL REQUEST chega à outra extremidade, a entidade de transporte é interrompida para executar *packet_arrival*, a fim de verificar se o usuário local está na escuta no endereço especificado. Se estiver, um pacote CALL ACCEPTED será enviado de volta, e o usuário remoto será despertado; caso contrário, a primitiva CALL REQUEST será enfileirada durante *TIMEOUT* pulsos do clock. Se uma primitiva LISTEN for executada nesse período, a conexão será estabelecida; caso contrário, ela entrará em timeout e será rejeitada com um pacote CLEAR REQUEST para que ela não fique bloqueada indefinidamente.

Embora tenhamos eliminado o cabeçalho do protocolo de transporte, ainda precisaremos de um meio para controlar qual pacote pertence a cada conexão de transporte, pois podem existir várias conexões simultâneas. A estratégia mais simples é usar o número do circuito virtual da camada de rede como número da conexão de transporte. Além disso, o número do circuito virtual também pode ser

usado como índice no array *conn*. Quando um pacote chega no circuito virtual *k* da camada de rede, ele pertence à conexão de transporte *k*, cujo estado se encontra no registro *conn*[*k*]. Para as conexões iniciadas em um host, o número da conexão é escolhido pela entidade de transporte de origem. Para as chamadas recebidas, a camada de rede opta por números de circuito virtual não utilizados.

[arte: ver original da p. 518-521]

[TD]

```
#define MAX_CONN 32     /* número máximo de conexões simultâneas */
#define MAX_MSG_SIZE 8192              /* maior mensagem em bytes */
#define MAX_PKT_SIZE 512               /* maior pacote em bytes */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT}
pkt_type;
typedef enum
{IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN} cstate;

/* Variáveis globais. */
transport_address listen_address;       /* endereço local que está sendo escutado
```

```
*/
int listen_conn;                      /* identificador de conexão para escuta */
unsigned char data[MAX_PKT_SIZE];     /* área de rascunho para dos de pacote */

struct conn {
  transport_address local_address, remote_address;
  cstate state;                               /* estado desta conexão */
  unsigned char *user_buf_addr;         /* ponteiro para buffer de recepção */
  int byte_count;                     /* contagem de envio/recepção */
  int clr_req_received;                       /* ativado ao ser recebido pacote
CLEAR_REQ */
  int timer;                          /* usado para timeout de pacotes CALL_REQ */
  int credits;                        /* número de mensagens que podem ser
enviadas */
} conn[MAX_CONN + 1];                         /* o slot 0 não é usado */

void sleep(void);                     /* protótipos */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{ /* O usuário quer escutar uma conexão. Verifica se CALL_REQ já chegou. */
  int i, found = 0;

  for (i = 1; i <= MAX_CONN; i++)       /* pesquisa na tabela em busca de
CALL_REQ */
```

```
        if (conn[i].state == QUEUED && conn[i].local_address == t) {
                found = i;
                break;
        }

  if (found == 0) {
        /* Nenhuma CALL_REQ aguardando. Fica suspenso até a chegada ou até
timeout. */
        listen_address = t;  sleep();  i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;           /* a conexão é estabelecida (ESTABLISHED)
*/
  conn[i].timer = 0;                /* o timer não é usado */
  listen_conn = 0;                  /* 0 é considerado um endereço inválido */
  to_net(i, 0, 0, CALL_ACC, data, 0);    /* diz à rede para aceitar a conexão */
  return(i);                        /* retorna identificador da conexão */
}

int connect(transport_address l, transport_address r)
{ /* O usuário deseja se conectar a um processo remoto; envia pacote CALL_REQ.
*/
  int i;
  struct conn *cptr;

  data[0] = r;   data[1] = l;                /* o pacote CALL_REQ precisa desses itens
*/
  i = MAX_CONN;                          /* pesquisa tabela em ordem inversa */
```

```
  while (conn[i].state != IDLE && i > 1) i = i - 1;

  if (conn[i].state == IDLE) {

       /* Cria uma entrada de tabela indicando que CALL_REQ foi ligado. */

       cptr = &conn[i];

       cptr->local_address = l; cptr->remote_address = r;

       cptr->state = WAITING; cptr->clr_req_received = 0;

       cptr->credits = 0; cptr->timer = 0;

       to_net(i, 0, 0, CALL_REQ, data, 2);

       sleep();                /* espera por CALL_ACC ou CLEAR_REQ */

       if (cptr->state == ESTABLISHED) return(i);

       if (cptr->clr_req_received) {

              /* O outro lado recusou a chamada. */

              cptr->state = IDLE;   /* volta ao estado IDLE */

              to_net(i, 0, 0, CLEAR_CONF, data, 0);

              return(ERR_REJECT);

       }

  } else return(ERR_FULL);          /* rejeita CONNECT: não há espaço nas tabelas */

}


int send(int cid, unsigned char bufptr[], int bytes)

{ /* O usuário deseja enviar uma mensagem. */

  int i, count, m;

  struct conn *cptr = &conn[cid];


  /* Entra no estado SENDING. */

  cptr->state = SENDING;

  cptr->byte_count = 0;                  /* # bytes enviados até esta mensagem */
```

```
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
      /* Crédito disponível; divide mensagem em pacotes, se necessário. */
      do {
          if (bytes - cptr->byte_count > MAX_PKT_SIZE) {        /* mensagem de
vários pcotes */
              count = MAX_PKT_SIZE;  m = 1;      /* mais pacote em seguida */
          } else {            /* mensagem com um único pacote */
              count = bytes - cptr->byte_count;  m = 0; /* último pacote desta
mensagem */
          }
          for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
          to_net(cid, 0, m, DATA_PKT, data, count);        /* envia um pacote */
          cptr->byte_count = cptr->byte_count + count;        /* incrementa
bytes enviados até agora */
      } while (cptr->byte_count < bytes); /* entra em loop até enviar a mensagem
inteira */
      cptr->credits--;               /* cada mensagem utiliza um crédito */
      cptr->state = ESTABLISHED;
      return(OK);
  } else {
      cptr->state = ESTABLISHED;
      return(ERR_CLOSED);                  /* a transmissão falhou: o par quer se
desconectar */
  }
}
```

```
int receive(int cid, unsigned char bufptr[], int *bytes)
{ /* O usuário está preparado para receber uma mensagem. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
       /* Conexão ainda estabelecida; tenta receber. */
       cptr->state = RECEIVING;
       cptr->user_buf_addr = bufptr;
       cptr->byte_count = 0;
       data[0] = CRED;
       data[1] = 1;
       to_net(cid, 1, 0, CREDIT, data, 2); /* envia crédito */
       sleep();                  /* bloqueia enquanto aguarda dados */
       *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
{ /* O usuário quer encerrar uma conexão. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {             /* o outro lado iniciou o processo de
término */
       cptr->state = IDLE;          /* agora a conexão é encerrada */
       to_net(cid, 0, 0, CLEAR_CONF, data, 0);
```

```
  } else {                              /* iniciamos o processo de término */
        cptr->state = DISCONN;          /* não é encerrada até o outro lado
concordar */
        to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}

void packet_arrival(void)
{ /* Chegou um pacote: recebe e processa o pacote. */
  int cid;                              /* conexão na qual o pacote chegou */
  int count, i, q, m;
  pkt_type ptype;     /* CALL_REQ, CALL_ACC, CLEAR_REQ, CLEAR_CONF,
DATA_PKT, CREDIT */
  unsigned char data[MAX_PKT_SIZE];   /* porção de dados do pacote de entrada
*/
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count);      /* recebe o pacote */
  cptr = &conn[cid];

  switch (ptype) {
    case CALL_REQ:                      /* o usuário remoto quer estabelecer uma
conexão */
        cptr->local_address = data[0];  cptr->remote_address = data[1];
        if (cptr->local_address == listen_address) {
              listen_conn = cid;  cptr->state = ESTABLISHED;  wakeup();
```

```
    } else {
        cptr->state = QUEUED;  cptr->timer = TIMEOUT;
    }
    cptr->clr_req_received = 0;   cptr->credits = 0;
    break;

  case CALL_ACC:                    /* o usuário remoto aceitou nosso
CALL_REQ */
    cptr->state = ESTABLISHED;
    wakeup();
    break;

  case CLEAR_REQ:                   /* o usuário remoto que desconectar ou
rejeitar chamada */
    cptr->clr_req_received = 1;
    if (cptr->state == DISCONN) cptr->state = IDLE;     /* resolve colisão */
    if (cptr->state ==  WAITING || cptr->state == RECEIVING || cptr->state ==
SENDING) wakeup();
    break;

  case CLEAR_CONF:                  /* o usuário remoto concorda em se
desconectar */
    cptr->state = IDLE;
    break;

  case CREDIT:                 /* o usuário remoto está aguardando dados */
    cptr->credits += data[1];
```

```
        if (cptr->state == SENDING) wakeup();
        break;

     case DATA_PKT:                    /* o usuário remoto enviou dados */
        for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] =
data[i];
        cptr->byte_count += count;
        if (m == 0 ) wakeup();
  }
}
void clock(void)
{ /* Houve um pulso do clock; verifica timeouts de solicitações de conexões
enfileiradas. */
  int i;
  struct conn *cptr;
  for (i = 1; i <= MAX_CONN; i++) {
        cptr = &conn[i];
        if (cptr->timer > 0) {         /* o timer estava funcionando */
                cptr->timer--;
                if (cptr->timer == 0) {   /* o timer expirou */
                        cptr->state = IDLE;
                        to_net(i, 0, 0, CLEAR_REQ, data, 0);
                }
        }
  }
} [TN]
```

[F]Figura 6.20

[FL] Um exemplo de entidade de transporte

Para evitar a necessidade de fornecer e gerenciar buffers dentro da entidade de transporte, utilizamos nesse caso um mecanismo de controle de fluxo diferente da janela deslizante normal. Quando um usuário chama uma primitiva RECEIVE, uma **mensagem de crédito** especial é enviada à entidade de transporte da máquina transmissora e é gravada no array *conn*. Quando uma primitiva SEND é chamada, a entidade de transporte verifica se chegou um crédito na conexão especificada. Caso isso tenha ocorrido, a mensagem será enviada (em vários pacotes, se necessário) e o crédito será decrementado; caso contrário, a entidade de transporte se colocará em estado suspenso até a chegada de um crédito. Esse mecanismo garante que nenhuma mensagem será enviada a menos que o outro lado já tenha executado uma primitiva RECEIVE. Como resultado, sempre que uma mensagem chegar, haverá um buffer disponível para ela. O esquema pode ser generalizado com facilidade, a fim de permitir que os receptores forneçam vários buffers e solicitem muitas mensagens.

Você deve ter em mente a simplicidade da Figura 6.20. Uma entidade de transporte real normalmente verificaria a validade de todos os parâmetros fornecidos, cuidaria da recuperação do funcionamento após uma falha na camada de rede, trataria das colisões de chamadas e seria compatível com um serviço de transporte mais genérico, com interrupções, datagramas e versões sem bloqueio das primitivas SEND e RECEIVE.

[T3] 6.3.3 O exemplo sob a forma de máquina de estados finitos

A elaboração do código de uma entidade de transporte é difícil e trabalhosa, especialmente para protocolos mais realistas. Para diminuir as chances de cometer erros, em geral é uma boa idéia representar o estado do protocolo como

uma máquina de estados finitos.

Já vimos que nosso exemplo de protocolo tem sete estados por conexão.

Também é possível isolar 12 eventos que podem fazer com que o estado de uma conexão se altere. Cinco desses eventos são as cinco primitivas de serviço. Outros seis são as chegadas dos seis tipos de pacotes válidos. O último é a expiração do timer. A Figura 6.21 mostra as ações do protocolo principal na forma de matriz. As colunas são os estados e as linhas representam os 12 eventos.

Cada entrada da matriz (ou seja, da máquina de estados finitos) da Figura 6.21 tem até três campos: um predicado, uma ação e um novo estado. O predicado indica sob quais circunstâncias a ação é executada. Por exemplo, na entrada superior esquerda, se uma ação LISTEN for executada e não houver mais espaço na tabela (predicado *P*1), LISTEN falhará e o estado não será alterado. Por outro lado, se um pacote CALL REQUEST já tiver chegado para o endereço de transporte que está sendo escutado (predicado *P*2), a conexão será imediatamente estabelecida. Outra possibilidade é *P*2 ser falso, ou seja, não chegar nenhum pacote CALL REQUEST; nesse caso, a conexão será mantida no estado *IDLE*, aguardando um pacote CALL REQUEST.

[arte: ver original p. 523]

[Dísticos]

[1]Primitivas

Pacotes recebidos

Clock

[2]LISTEN

CONNECT

DISCONNECT

SEND

A12/Estab

A11/Estab A7/Estab

~/Idle

[5]

Predicados

P1: Tabela de conexão cheia

P2: Call_req pendente

P3: LISTEN pendente

P4: Clear_req pendente

P5: Crédito disponível

Ações

A1: Enviar Call_acc

A2: Esperar por Call_req

A3: Enviar Call_req

A4: Ativar timer

A5: Enviar Clear_conf

A6: Enviar Clear_req

A7: Enviar mensagem

A8: Esperar por crédito

A9: Enviar crédito

A10: Ativar flag Clr_req_received

A11: Registrar crédito

A12: Aceitar mensagem

[F]Figura 6.21

[FL] O exemplo de protocolo como uma máquina de estados finitos. Cada entrada tem um predicado opcional, uma ação opcional e um novo estado. O til (~) indica que nenhuma ação mais importante foi executada. Um traço sobre um predicado

indica a negação do predicado. As entradas em branco correspondem a eventos impossíveis ou inválidos

Vale a pena ressaltar que a escolha dos estados a serem usados na matriz não é totalmente limitada pelo protocolo. Nesse exemplo, não há estado *LISTENING*, o que pode ser razoável após LISTEN. Não há um estado *LISTENING*, pois um estado está sempre associado a uma entrada de registro de conexão, e nenhum registro de conexão é criado por LISTEN. Por que não? Porque decidimos usar os números dos circuitos virtuais da camada de rede como identificadores da conexão e, para uma ação LISTEN, o número do circuito virtual é escolhido pela camada de transporte quando chega o pacote CALL REQUEST.

As ações de *A*1 a *A*12 são as principais, tais como enviar pacotes ou acionar os timers. Nem todas as ações secundárias, tais como inicializar os campos de um registro de conexão, estão listadas. Se uma ação envolver despertar um processo suspenso, as ações seguintes também serão levadas em conta. Por exemplo, se um pacote CALL REQUEST chegar e um processo estiver em suspenso aguardando o pacote, a transmissão do pacote CALL ACCEPT subseqüente ao despertar do processo contará como parte da ação CALL REQUEST. Depois de cada ação ser executada, a conexão poderá passar para um novo estado, como ilustra a Figura 6.21.

A vantagem de representar o protocolo como uma matriz é tríplice. Primeiro, nesse formato fica muito mais fácil para o programador verificar sistematicamente cada combinação de estado e evento, para constatar se alguma ação é necessária. Em implementações comerciais, algumas dessas combinações seriam usadas para o tratamento de erros. Na Figura 6.21 não há distinção entre situações impossíveis e inválidas. Por exemplo, se uma conexão estiver no estado *waiting*, o evento DISCONNECT será impossível, pois o usuário será bloqueado e

não poderá executar qualquer primitiva. Por outro lado, no estado *sending* não são aguardados pacotes de dados, pois nenhum crédito foi transmitido. A chegada de um pacote de dados é um erro do protocolo.

A segunda vantagem da representação do protocolo por uma matriz está em sua própria implementação. É possível imaginar um array bidimensional no qual o elemento *a*[*i*][*j*] seria um ponteiro ou índice para o procedimento que trataria da ocorrência do evento *i* no estado *j*. Uma implementação viável é escrever a entidade de transporte como um pequeno loop, esperando por um evento no início do loop. Quando ocorrer um evento, a conexão correspondente será localizada e seu estado extraído. Com o evento e o estado conhecidos, a entidade de transporte apenas indexará o array *a* e chamará o procedimento apropriado. Essa estratégia gera um projeto muito mais regular e sistemático do que a nossa entidade de transporte.

A terceira vantagem da estratégia de máquina de estados finitos diz respeito à descrição de protocolos. Em alguns documentos de padrões, os protocolos são fornecidos como máquinas de estados finitos do tipo da Figura 6.21. Sair desse tipo de descrição e passar para uma entidade de transporte que funcione será muito mais fácil se a entidade de transporte também estiver orientada por uma máquina de estados finitos baseada na máquina descrita no padrão.

A principal desvantagem da estratégia de máquina de estados finitos é que ela pode ser mais difícil de entender que o exemplo de programação direta que utilizamos inicialmente. Contudo, esse problema pode ser resolvido em parte desenhando-se a máquina de estados finitos sob a forma de um grafo, como foi feito na Figura 6.22.

[arte: ver original p. 525]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja

digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 6.22

[FL] O exemplo de protocolo em forma de grafo. Para tornar o processo mais simples, foram omitidas as transições que não alteram o estado da conexão

[T2] 6.4 Os protocolos de transporte da Internet: UDP

A Internet tem dois protocolos principais na camada de transporte, um protocolo sem conexões e outro orientado a conexões. Nas próximas seções, estudaremos os dois. O protocolo sem conexões é o UDP. O protocolo orientado a conexões é o TCP. Como o UDP é basicamente o IP com um pequeno cabeçalho, vamos começar por ele. Também examinaremos duas aplicações do UDP.

[T3] 6.4.1 Introdução ao UDP

O conjunto de protocolos da Internet admite um protocolo de transporte sem conexões, o **UDP (User Datagram Protocol)**. O UDP oferece um meio para as aplicações enviarem datagramas IP encapsulados sem que seja necessário estabelecer uma conexão. O UDP é descrito na RFC 768.

O UDP transmite **segmentos** que consistem em um cabeçalho de 8 bytes, seguido pela carga útil. O cabeçalho é mostrado na Figura 6.23. As duas portas servem para identificar os pontos extremos nas máquinas de origem e destino. Quando um pacote UDP chega, sua carga útil é entregue ao processo associado à porta de destino. Essa associação ocorre quando a primitiva BIND ou algo semelhante é usado, como vimos na Figura 6.6 para o TCP (o processo de vinculação é idêntico para o UDP). De fato, o principal valor de se ter o UDP em relação ao uso do IP bruto é a adição das portas de origem e destino. Sem os campos de portas, a camada de transporte não saberia o que fazer com o pacote. Com eles, a camada entrega segmentos corretamente.

[arte: ver original p. 526]

[Dísticos]

[1]32 Bits

[2]Source port Destination port

[3]UDP length UDP checksum

[F]Figura 6.23

[FL] O cabeçalho do UDP

A porta de origem é usada principalmente quando uma resposta dever ser devolvida à origem. Copiando o campo *Source port* do segmento de entrada no campo *Destination port* do segmento de saída, o processo que transmite a resposta pode especificar qual processo na máquina transmissora deve recebê-lo. O campo *UDP length* inclui o cabeçalho de 8 bytes e os dados. O campo *UDP checksum* é opcional e armazenado como 0 se não for calculado (um valor 0 verdadeiro calculado é armazenado com todos os bits iguais a 1). É tolice desativá-lo, a menos que a qualidade dos dados não tenha importância (por exemplo, no caso de voz digitalizada).

Vale a pena mencionar explicitamente algumas ações que o UDP *não* realiza. Ele não realiza controle de fluxo, controle de erros ou retransmissão após a recepção de um segmento incorreto. Tudo isso cabe aos processos do usuário. O que ele faz é fornecer uma interface para o protocolo IP com o recurso adicional de demultiplexação de vários processos que utilizam as portas. Isso é tudo que ele faz. Para aplicações que precisam ter controle preciso sobre o fluxo de pacotes, o controle de erros ou a sincronização, o UDP fornece apenas aquilo que é determinado.

Uma área na qual o UDP é especialmente útil é a de situações cliente/servidor. Com freqüência, o cliente envia uma pequena solicitação ao servidor e espera

uma pequena resposta de volta. Se a solicitação ou a resposta se perder, o cliente simplesmente chegará ao timeout e tentará de novo. Não só o código é simples, mas é necessário um número menor de mensagens (uma em cada sentido) do que no caso de um protocolo que exige uma configuração inicial.

Uma aplicação que utiliza o UDP desse modo é o DNS (Domain Name System), que estudaremos no Capítulo 7. Em resumo, um programa que precisa pesquisar o endereço IP de algum nome de host — por exemplo, www.cs.berkeley.edu — pode enviar um pacote UDP contendo o nome do host a um servidor DNS. O servidor responde com um pacote UDP que contém o endereço IP. Não é necessária nenhuma configuração antecipada e também nenhum encerramento posterior. Basta enviar duas mensagens pela rede.

[T3] 6.4.2 Chamada de procedimentos remotos

Em um certo sentido, enviar uma mensagem a um host remoto e obter de volta uma resposta é muito semelhante a criar uma chamada de função em uma linguagem de programação. Em ambos os casos, você começa com um ou mais parâmetros e recebe de volta um resultado. Essa observação levou as pessoas a tentarem organizar interações de solicitação/resposta em redes no formato de chamadas de procedimentos. Tal organização torna as aplicações de rede muito mais fáceis de programar e mais familiares. Por exemplo, imagine um procedimento chamado *get_IP_address* (*host_name*) que funciona enviando um pacote UDP a um servidor DNS e aguardando a resposta, chegando ao timeout e tentando de novo, caso não receba uma resposta com rapidez suficiente. Desse modo, todos os detalhes de redes podem ficar ocultos do programador.

O trabalho fundamental nessa área foi realizado por Birrell e Nelson (1984). Em resumo, o que Birrell e Nelson sugeriram foi permitir que os programas chamassem procedimentos localizados em hosts remotos. Quando um processo

na máquina 1 chama um procedimento na máquina 2, o processo de chamada em 1 é suspenso, e a execução do procedimento chamado ocorre em 2. As informações podem ser transportadas do chamador até o chamado nos parâmetros, e pode voltar no resultado do procedimento. Nenhuma passagem de mensagens é visível para o programador. Essa técnica é conhecida como **RPC (Remote Procedure Call — chamada de procedimento remoto)** e se tornou a base para muitas aplicações de redes. Tradicionalmente, o procedimento chamador é conhecido como cliente, e o procedimento chamado é conhecido como servidor; também usaremos esses nomes aqui.

A idéia por trás da RPC é tornar uma chamada de procedimento remoto o mais semelhante possível a uma chamada local. Na forma mais simples, para chamar um procedimento remoto, o programa cliente deve estar vinculado a um pequeno procedimento de biblioteca, chamado **stub do cliente**, que representa o procedimento do servidor no espaço de endereços do cliente. De modo semelhante, o servidor está vinculado a um procedimento chamado **stub do servidor.** Esses procedimentos ocultam o fato de que a chamada de procedimento do cliente até o servidor não é local.

As etapas reais na criação de uma RPC são mostrados na Figura 6.24. A etapa 1 é a chamada do cliente ao stub do cliente. Essa é uma chamada de procedimento local, com os parâmetros colocados na pilha da maneira normal. A etapa 2 é o stub do cliente reunindo os parâmetros em uma mensagem e efetuando uma chamada de sistema para enviar a mensagem. A reunião dos parâmetros é chamada **empacotamento (marshaling)**. A etapa 3 é o núcleo enviando a mensagem da máquina cliente até a máquina servidora. A etapa 4 é o núcleo passando o pacote recebido ao stub do servidor. Finalmente, a etapa 5 é o stub do servidor chamando o procedimento servidor com os desempacotados. A resposta segue o mesmo caminho no sentido inverso.

O principal detalhe que devemos observar nesse caso é que o procedimento cliente, escrito pelo usuário, simplesmente realiza uma chamada de procedimento normal (isto é, local) ao stub do cliente, que tem o mesmo nome que o procedimento servidor. Tendo em vista que o procedimento cliente e stub do cliente estão no mesmo espaço de endereços, os parâmetros são repassados no modo habitual. De forma semelhante, o procedimento servidor é chamado por um procedimento em seu espaço de endereços com os parâmetros que espera. Para o procedimento servidor, nada é incomum. Desse modo, em vez de ser realizada a E/S em soquetes, a comunicação de rede é feita simulando-se uma chamada de procedimento normal.

Apesar da elegância conceitual da RPC, existem alguns períodos ocultos. Um deles é o uso de parâmetros de ponteiros. Normalmente, a passagem de um ponteiro a um procedimento não é problema. O procedimento chamado pode usar um ponteiro do mesmo modo que o chamador o utiliza, porque ambos os procedimentos convivem no mesmo espaço de endereço virtuais. Com a RPC, a passagem de ponteiros é impossível, porque o cliente e o servidor estão em espaços de endereços diferentes.

[arte: ver original p. 528]

[Dísticos]

[1] CPU do cliente

1 Stub do cliente

Cliente

2

Sistema operacional

3

[2] CPU do servidor

Stub do servidor 5

Servidor

4

Sistema operacional

Rede

[F]Figura 6.24

[FL] Etapas na criação de uma chamada de procedimento remoto. Os stubs estão sombreados

Em alguns casos, podem ser usados artifícios para torna possível a passagem de ponteiros. Suponha que o primeiro parâmetro seja um ponteiro para um inteiro *k*. O stub do cliente pode empacotar *k* e enviá-lo para o servidor. O stub do servidor cria então um ponteiro para *k* e o repassa ao procedimento servidor, da maneira esperada. Quando o procedimento servidor devolve o controle ao stub do servidor, este último envia *k* de volta ao cliente, onde o novo *k* é copiado sobre o antigo, pois o servidor pode tê-lo alterado. Na realidade, a seqüência de chamada padrão da chamada por referência foi substituída pela cópia/restauração. Infelizmente, esse artifício nem sempre funciona; por exemplo, se o ponteiro indicar um grafo ou outra estrutura de dados complexa. Por essa razão, algumas restrições devem ser impostas sobre parâmetros para procedimentos chamados remotamente.

Um segundo problema é que, em linguagens com tipificação fraca como C, é perfeitamente válido escrever um procedimento que calcula o produto interno de dois vetores (arrays), sem especificar o tamanho de cada vetor. Cada um deles poderia terminar com um valor especial conhecido apenas pelo procedimento de chamada e pelo procedimento chamado. Sob essas circunstâncias, é essencialmente impossível para o stub do cliente empacotar os parâmetros: ele não tem como determinar o tamanho desses parâmetros.

Um terceiro problema é que nem sempre é possível deduzir os tipos dos parâmetros, nem mesmo a partir de uma especificação formal ou do próprio código. Um exemplo é *printf*, que pode ter qualquer número de parâmetros (pelo menos um), e os parâmetros podem ser uma mistura arbitrária de números inteiros, curtos, longos, de caracteres, de strings, de números em ponto flutuante de diversos tamanhos e de outros tipos. Tentar chamar *prinif* como um procedimento remoto seria praticamente impossível, porque C é uma linguagem muito permissiva. Porém, uma regra estabelecendo que a RPC pode ser usada desde que você não programe em C (ou em C++) não seria muito popular.

Um quarto problema se relaciona ao uso de variáveis globais. Normalmente, o procedimento chamador e o procedimento chamado podem se comunicar usando variáveis globais, além de parâmetros. Se o procedimento chamado for agora deslocado para uma máquina remota, o código falhará, porque as variáveis globais não serão mais compartilhadas.

Esses problemas não pretendem sugerir que a RPC é impossível. De fato, ela é amplamente utilizada, mas são necessárias algumas restrições para fazê-la funcionar bem na prática.

É claro que a RPC não precisa usar pacotes UDP, mas RPC e UDP se adaptam bem, e o UDP é comumente utilizado para RPC. Porém, quando os parâmetros ou resultados são maiores que o pacote máximo do UDP, ou quando a operação solicitada não é idempotente (isto é, não pode ser repetida com segurança, como ocorre quando se incrementa um contador), pode ser necessário instalar uma conexão TCP e enviar a solicitação por ela, em vez de usar o UDP.

[T3] 6.4.3 O RTP (Real-time Transport Protocol)

A RPC do tipo cliente/servidor é uma área em que o UDP é amplamente utilizado. Outra área é a das aplicações de multimídia em tempo real. Em particular, à

medida que o rádio da Internet, a telefonia da Internet, a música por demanda, a videoconferência, o vídeo sob demanda e outras aplicações de multimídia se tornaram mais comuns, as pessoas descobriram que cada aplicação estava reinventando aproximadamente o mesmo protocolo de transporte em tempo real. aos poucos, ficou claro que seria uma boa idéia ter um protocolo de transporte de tempo real genérico para várias aplicações. Desse modo, foi criado o **RTP (Real-time Transport Protocol).** Ele é descrito na RFC 1889 e agora está em uso difundido.

A posição do RTP na pilha de protocolos é um pouco estranha. Decidiu-se que ele deveria ser inserido no espaço do usuário e, desse modo, ser (normalmente) executado sobre o UDP. O RTP opera da maneira descrita a seguir. A aplicação de multimídia consiste em vários fluxos de áudio, vídeo, texto e possivelmente outros fluxos. Esses fluxos são armazenados na biblioteca RTP, que se encontra no espaço do usuário, juntamente com a aplicação. Essa biblioteca efetua a multiplexação dos fluxos e os codifica em pacotes RTP, que são então colocados em um soquete. Na outra extremidade do soquete (no núcleo do sistema operacional), os pacotes UDP são gerados e incorporados a pacotes IP. Se o computador estiver em uma rede Ethernet, os pacotes IP serão inseridos em quadros Ethernet para transmissão. A pilha de protocolos para essa situação é mostrada na Figura 6.25(a). O aninhamento de pacotes é mostrado na Figura 6.25(b).

[arte: ver original p. 529]

[Dísticos]

[1] Espaço do usuário Aplicação de multimídia

RTP

Interface de soquete

Núcleo do SO UDP

IP

Ethernet

(a)

[2] Cabeçalho Ethernet Cabeçalho IP Cabeçalho UDP Cabeçalho RTP

Carga útil do RTP

Carga útil do UDP

Carga útil do IP

Carga útil Ethernet

(b)

[F]Figura 6.25

[FL] (a) A posição do RTP na pilha de protocolos. (b) O aninhamento de pacotes

Como conseqüência dessa estrutura, é um pouco difícil dizer em que camada o RTP está. Como ele funciona no espaço do usuário e está vinculado ao programa aplicativo, certamente parece ser um protocolo de aplicação. Por outro lado, ele é um protocolo genérico e independente das aplicações que apenas fornecem recursos de transporte, e assim também é semelhante a um protocolo de transporte. Talvez a melhor descrição do RTP seja como um protocolo de transporte implementado na camada de aplicação.

A função básica do RTP é multiplexar diversos fluxos de dados de tempo real sobre um único fluxo de pacotes UDP. O fluxo UDP pode ser enviado a um único destino (unidifusão) ou a vários destinos (multidifusão). Como o RTP utiliza simplesmente o UDP normal, seus pacotes não são tratados de maneira especial pelos roteadores, a menos que alguns recursos de qualidade de serviço normais do IP estejam ativos. Em particular, não há nenhuma garantia especial sobre entrega, flutuação etc.

Cada pacote enviado em um fluxo RTP recebe um número uma unidade maior que

seu predecessor. Essa numeração permite ao destino descobrir se algum pacote está faltando. Se um pacote for omitido, a melhor ação que o destino deve executar é fazer a aproximação do valor que falta por interpolação. A retransmissão não é uma opção prática, pois o pacote retransmitido provavelmente chegaria tarde demais para ser útil. Como conseqüência, o RTP não tem nenhum controle de fluxo, nenhum controle de erros, nenhuma confirmação e nenhum mecanismo para solicitar retransmissões.

Cada carga útil do RTP pode conter várias amostras, e elas podem ser codificados de qualquer forma desejada pela aplicação. Para permitir a interoperação, o RTP define vários perfis (por exemplo, um único fluxo de áudio) e, para cada perfil, podem ser permitidos vários formatos de codificação. Por exemplo, um único fluxo de áudio pode ser codificado como amostras PCM de 8 bits a 8 kHz, codificação delta, codificação profética, codificação GSM, MP3 e assim por diante. O RTP fornece um campo de cabeçalho no qual a origem pode especificar a codificação, mas que não tem nenhum outro envolvimento na maneira de realizar a codificação.

Outro recurso de que muitas aplicações em tempo real necessitam é a marcação com timbre de hora. Aqui, a idéia é permitir que a origem associe um timbre de hora com a primeira amostra em cada pacote. Os timbres de hora são relativos ao início do fluxo, e assim somente as diferenças entre os timbres de hora são significativas. Os valores absolutos não têm nenhum significado. Esse mecanismo permite ao destino realizar alguma bufferização e reproduzir cada amostra depois de um número correto de milissegundos, contados desde o início do fluxo, independente de quando chegou o pacote contendo a amostra. O uso de timbres de hora não apenas reduz os efeitos da flutuação, mas também permite a sincronização de vários fluxos. Por exemplo, um programa de televisão digital poderia ter um fluxo de vídeo e dois fluxos de áudio. Os dois fluxos de áudio

poderiam ser para difusões estereofônicas ou para tratamento de filmes com uma trilha sonora no idioma original e outra dublada no idioma local, dando ao espectador a possibilidade de escolher. Cada fluxo vem de um dispositivo físico diferente mas, se eles forem marcados com um timbre de hora baseado em um único contador, poderão ser reproduzidos de modo sincronizado, mesmo que os fluxos sejam transmitidos de maneira um tanto errática.

O cabeçalho do RTP é ilustrado na Figura 6.26. Ele consiste em três palavras de 32 bits e, potencialmente, algumas extensões. A primeira palavra contém o campo de *Version*, que já está em 2. Vamos esperar que essa versão esteja bem próxima da versão final, pois só falta definir um ponto de código (embora 3 talvez seja definido como uma indicação de que a versão real estava em uma palavra de extensão).

[arte: ver original p. 531]

[Dísticos]

[1] 32 bits

[2] Ver. P X CC M Payload type Sequence number

[3] Timestamp

[4] Synchronization source identifier

[5] Contributing source identifier

[F]Figura 6.26

[FL] O cabeçalho do RTP

O bit *P* indica que o pacote foi completado até chegar a um múltiplo de 4 bytes. O último byte de preenchimento informa quantos bytes foram acrescentados. O bit *X* indica que um cabeçalho de extensão está presente. O formato e o significado do cabeçalho de extensão não são definidos. O único detalhe definido é que a primeira palavra da extensão fornece o comprimento. Essa é uma válvula de

escape para quaisquer exigências imprevistas.

O campo *CC* informa quantas origens de contribuição estão presentes, de 0 a 15 (veja a seguir). O bit *M* é um bit marcador específico da aplicação. Ele pode ser usado para marcar o começo de um quadro de vídeo, o começo de uma palavra em um canal de áudio ou qualquer outro elemento que a aplicação reconheça. O campo *Payload type* informa que algoritmo de codificação foi usado (por exemplo, áudio não compactado de 8 bits, MP3 etc.). Tendo em vista que todo pacote apresenta esse campo, a codificação pode mudar durante a transmissão. O campo *Sequence number* é apenas um contador incrementado em cada pacote RTP enviado. Ele é usado para detectar pacotes pedidos.

O timbre de hora é produzido pela origem do fluxo para anotar quando a primeira amostra no pacote foi realizada. Esse valor pode ajudar a reduzir a flutuação no receptor, desacoplando a reprodução do momento da chegada do pacote. O *Synchronization source identifier* informa a que fluxo o pacote pertence. Esse é o método usado para multiplexar e demultiplexar vários fluxos de dados em um único fluxo de pacotes UDP. Finalmente, os campos *Contributing source identifiers*, se estiverem presentes, serão usados quando houver misturadores (mixers) de áudio no estúdio. Nesse caso, o misturador será a origem de sincronização, e os fluxos que estão sendo mixados serão listados nesse campo.

O protocolo RTP tem um irmão caçula, chamado **RTCP (Real-time Transport Control Protocol).** Ele cuida do feedback, da sincronização e da interface do usuário, mas não transporta quaisquer dados. A primeira função pode ser usada para fornecer feedback sobre retardo, flutuação, largura de banda, congestionamento e outras propriedades de rede para as origens. Essas informações podem ser usadas pelo processo de codificação para aumentar a taxa de dados (e oferecer melhor qualidade) quando a rede estiver funcionamento

bem e para reduzir a taxa de dados quando houver problemas na rede.
Fornecendo feedback contínuo, os algoritmos de codificação podem ser adaptados continuamente para oferecer a melhor qualidade possível sob as circunstâncias atuais. Por exemplo, se a largura de banda aumentar ou diminuir durante a transmissão, a codificação pode passar de MP3 para PCM de 8 bits e para codificação delta, conforme necessário. O campo de tipo *Payload type* é usado para informar ao destino qual algoritmo de codificação será empregado no pacote atual, tornando possível variar o algoritmo de acordo com a demanda.
O RTCP também lida com a sincronização entre fluxos. O problema é que diferentes fluxos podem utilizar clocks distintos, com granularidades e taxas de flutuação diferentes. O RTCP pode ser usado para manter esses elementos sincronizados.
Finalmente, o RTCP fornece um modo para nomear as diversas origens (por exemplo, em texto ASCII). Essas informações podem ser exibidas na tela do receptor, a fim de indicar quem está se comunicando no momento.
Para obter mais informações sobre o RTP, consulte (Perkins, 2002).

[T2] 6.5 Os protocolos de transporte da Internet: TCP

O UDP é um protocolo simples e tem alguns usos específicos, como interações cliente/servidor e multimídia; porém, para a maioria das aplicações da Internet, é necessária uma entrega confiável e em seqüência. O UDP não pode proporcionar isso, e assim foi preciso criar outro protocolo. Ele se chama TCP e é o principal elemento da Internet. Vamos estudá-lo em detalhes nas próximas seções.

[T3] 6.5.1 Introdução ao TCP

O **TCP (Transmission Control Protocol)** foi projetado especificamente para oferecer um fluxo de bytes fim a fim confiável em uma inter-rede não confiável.

Uma inter-rede é diferente de uma única rede porque suas diversas partes podem ter topologias, larguras de banda, retardos, tamanhos de pacote e outros parâmetros completamente diferentes. O TCP foi projetado para se adaptar dinamicamente às propriedades da inter-rede e ser robusto diante dos muitos tipos de falhas que podem ocorrer.

O TCP foi formalmente definido na RFC 793. Com o passar do tempo, vários erros e inconsistências foram detectados e muitos requisitos mudaram em algumas áreas. Esses esclarecimentos e as soluções de alguns bugs são descritos com detalhes na RFC 1122. As extensões são fornecidas na RFC 1323.

Cada máquina compatível com o TCP tem uma entidade de transporte TCP, que pode ser um procedimento de biblioteca, um processo do usuário ou parte do núcleo. em todos os casos, ele gerencia fluxos e interfaces TCP para a camada IP. Uma entidade TCP aceita fluxos de dados do usuário provenientes de processos locais, divide-os em partes de no máximo 64 Kb (na prática, com freqüência temos 1460 bytes de dados, para que ele possa caber em um único quadro Ethernet com os cabeçalhos IP e TCP) e envia cada parte em um datagrama IP distinto. Quando os datagramas IP que contêm dados TCP chegam a uma máquina, eles são enviados à entidade TCP, que restaura os fluxos de bytes originais. Para simplificar, às vezes utilizamos apenas "TCP", a fim de fazer referência tanto à entidade de transporte TCP (um software) quanto ao protocolo TCP (um conjunto de regras). Pelo contexto, ficará claro a qual deles estaremos nos referindo. Por exemplo, em "O usuário envia os dados para TCP", está claro que estamos nos referindo à entidade de transporte TCP.

A camada IP não oferece qualquer garantia de que os datagramas serão entregues da forma apropriada; portanto, cabe ao TCP administrar os timers e retransmiti-los sempre que necessário. Os datagramas também podem chegar fora de ordem; o TCP também terá de reorganizá-los em mensagens na seqüência correta.

Resumindo, o TCP deve fornecer a confiabilidade que a maioria dos usuários deseja, mas que o IP não oferece.

[T3] 6.5.2 O modelo de serviço do TCP

O serviço TCP é obtido quando tanto o transmissor quanto o receptor criam pontos extremos chamados soquetes, como vimos na Seção 6.1.3. Cada soquete tem um número de soquete (endereço) que consiste no endereço IP do host e em um número de 16 bits local para esse host, chamado **porta**. Porta é o nome usado pelo TCP para um TSAP. Para que o serviço TCP funcione, é necessário que uma conexão seja explicitamente estabelecida entre um soquete da máquina transmissora e um soquete da máquina receptora. As chamadas de soquetes estão listadas na Figura 6.5.

Um soquete pode ser utilizado por várias conexões ao mesmo tempo. Em outras palavras, duas ou mais conexões podem terminar no mesmo soquete. As conexões são identificadas nas duas extremidades pelos identificadores de soquetes, ou seja (*soquete*1,*soquete*2). Nenhum número de circuito virtual ou qualquer outro identificador é usado.

As portas com números abaixo de 1024 são denominadas **portas conhecidas** e são reservadas para serviços padrão. Por exemplo, qualquer processo que deseje estabelecer uma conexão com um host para transferir um arquivo usando FTP pode se conectar à porta 21 do host de destino e entrar em contato com seu daemon de FTP. A lista de portas conhecidas é dada em *www.iana.org*. Mais de 300 já foram atribuídas. algumas das mais conhecidas estão listadas na Figura 6.27.

Certamente seria possível fazer o daemon de FTP se associar à porta 21 durante a inicialização, fazer o daemon telnet se associar à porte 23 em tempo de inicialização e assim por diante. Porém, isso ocuparia a memória com daemons

que ficariam ociosos na maior parte do tempo. Em vez disso, geralmente se tem um único daemon, chamado **inetd (Internet daemon)** no UNIX que se associa a várias portas e espera pela primeira conexão de entrada. Quando isso ocorre, o inetd ativa um novo processo e executa nele o daemon apropriado, deixando esse daemon tratar a solicitação. Desse modo, os daemons diferentes de inetd só estão ativos quando há trabalho para eles realizarem. O inetd descobrem que porta devem usar em um arquivo de configuração. Conseqüentemente, o administrador do sistema pode configurá-lo de modo a ter daemons permanentes nas portas mais ocupadas (por exemplo, a porta 80) e inetd nas portas restantes.

[arte: ver original p. 534a]

[T]Tabela

Porta	Protocolo	Uso
21	FTP	Transferência de arquivos
23	Telnet	Login remoto
25	SMTP	Correio eletrônico
69	TFTP	Protocolo trivial de transferência de arquivos
79	Finger	Pesquisa de informações sobre um usuário
80	HTTP	World Wide Web
110	POP-3	Acesso remoto a correio eletrônico
119	NNTP	Notícias da USENET

[F]Figura 6.27

[FL] Algumas portas atribuídas

Todas as conexões TCP são full-duplex e ponto a ponto. Full-duplex quer dizer que o tráfego pode ser feito em ambas as direções ao mesmo tempo. Ponto a ponto significa que cada conexão possui exatamente dois pontos terminais. O TCP não admite os processos de multidifusão e difusão.

Uma conexão TCP é um fluxo de bytes e não um fluxo de mensagens. As fronteiras das mensagens não são preservadas de uma extremidade à outra. Por exemplo, se o processo transmissor executar quatro gravações de 512 bytes em um fluxo TCP, esses dados poderão ser entregues ao processo receptor em quatro partes de 512 bytes, em duas de 1024 bytes, uma de 2048 bytes (ver Figura 6.28) ou em qualquer outra divisão. Não há um meio do receptor detectar a(s) unidade(s) em que os dados foram gravados.

[arte: ver original p. 534b]

[Dísticos]

[1]Cabeçalho IP Cabeçalho TCP

A B C D

(a)

[2]A B C D

(b)

[F]Figura 6.28

[FL] (a) Quatro segmentos de 512 bytes enviados como datagramas IP separados. (b) Os 2048 bytes de dados entregues à aplicação em uma única chamada READ

No UNIX, os arquivos também têm essa propriedade. O leitor de um arquivo não é capaz de distinguir se ele foi gravado um bloco por vez, um byte por vez ou todo de uma vez. A exemplo do que acontece com um arquivo UNIX, o software TCP não tem idéia do significado dos bytes, e também não está interessado em descobri-lo. Um byte é apenas um byte.

Quando uma aplicação repassa dados para a entidade TCP, ela pode enviá-los imediatamente ou armazená-los em um buffer (para aguardar outros dados e enviar um volume maior de uma só vez), de acordo com suas necessidades. Entretanto, há ocasiões em que a aplicação realmente quer que os dados sejam

enviados de imediato. Por exemplo, suponha que um usuário tenha se conectado a uma máquina remota. Depois que uma linha de comandos é preenchida e a tecla Enter (ou Carriage Return) é pressionada, é essencial que a linha seja transportada à máquina remota imediatamente, e não guardada no buffer até a chegada da próxima linha. Para forçar a saída dos dados, as aplicações podem usar o flag PUSH, que informa ao serviço TCP para não retardar a transmissão. Algumas aplicações antigas utilizavam o flag PUSH como um tipo de marcador para assinalar os limites das mensagens. Ainda que esse artifício às vezes funcione, ele pode falhar, pois nem todas as implementações do TCP repassam o flag PUSH para a aplicação no lado receptor. Além disso, se outros flags PUSH chegarem antes da transmissão do primeiro (por exemplo, porque a linha de saída está ocupada), o serviço TCP terá a liberdade de agrupar todos os dados que contiverem o flag PUSH em um único datagrama IP, sem qualquer separação entre as várias partes.

Um último recurso do serviço TCP que vale a pena mencionar é o dos **dados urgentes**. Quando um usuário interativo pressiona a tecla DEL ou as teclas CTRL-C para interromper um processo remoto já iniciado, a aplicação transmissora adiciona algumas informações de controle ao fluxo de dados e o entrega ao TCP juntamente com um flag URGENT. Isso faz com que o serviço TCP pare de acumular dados e transmita tudo imediatamente. Quando os dados urgentes são recebidos no destino, a aplicação receptora é interrompida (na terminologia UNIX, ela recebe um sinal) e pára tudo o que estiver fazendo para ler o fluxo de dados e encontrar os dados urgentes. O final dos dados urgentes é marcado para que a aplicação saiba quando eles terminarem. O início dos dados urgentes não é marcado, e a aplicação deve saber identificá-lo. Basicamente, esse esquema oferece um mecanismo de sinalização pouco sofisticado, deixando a maior parte do trabalho para a aplicação.

[T3] 6.5.3 O protocolo TCP

Nesta seção, apresentaremos uma visão geral do protocolo TCP. Na próxima seção, veremos o cabeçalho do protocolo, campo a campo.

Uma característica fundamental do TCP que domina o projeto do protocolo é que cada byte em uma conexão TCP tem seu próprio número de seqüência de 32 bits. Quando a Internet começou, as linhas entre roteadores eram principalmente linhas dedicadas de 56 kbps, e assim um host funcionando a toda velocidade demorava mais de uma semana para percorrer todos os números de seqüência. Nas velocidades das redes modernas, os números de seqüência podem ser consumidos a uma taxa alarmante, como veremos mais adiante. São usados números de seqüência de 32 bits separados para confirmações e para o mecanismo de janelas, como descreveremos a seguir.

As entidades transmissoras e receptoras do TCP trocam dados na forma de segmentos. Um **segmento TCP** consiste em um cabeçalho fixo de 20 bytes (além de uma parte opcional), seguido por zero ou mais bytes de dados. O software TCP decide qual deve ser o tamanho dos segmentos. Ele pode acumular dados de várias gravações em um único segmento ou dividir os dados de uma única gravação em vários segmentos. Dois fatores restringem o tamanho do segmento. Primeiro, cada segmento, incluindo o cabeçalho do TCP, deve caber na carga útil do IP, que é de 65.515 bytes. Em segundo lugar, cada rede tem uma **unidade máxima de transferência**, ou **MTU (Maximum Transfer Unit)** e cada segmento deve caber na MTU. Na prática, a MTU geralmente tem 1500 bytes (o tamanho da carga útil Ethernet) e, portanto, define o limite superior de tamanho de segmentos.

O protocolo básico utilizado pelas entidades TCP é o protocolo de janela deslizante. Quando envia um segmento, o transmissor também dispara um timer.

Quando o segmento chega ao destino, a entidade TCP receptora retorna um segmento (com ou sem dados, de acordo com as circunstâncias) com um número de confirmação igual ao próximo número de seqüência que espera receber. Se o timer do transmissor expirar antes da confirmação ser recebida, o segmento será retransmitido.

Apesar desse protocolo parecer simples, há muitos detalhes sobre ele que veremos a seguir. Os segmentos podem chegar fora de ordem; assim, os bytes 3072 a 4095 podem chegar, mas não podem ser confirmados, porque os bytes 2048 a 3071 ainda não chegaram. Além disso, os segmentos podem se atrasar tanto que o timer do transmissor expira e ele tem de retransmiti-los. As retransmissões podem incluir diferentes faixas de bytes em relação à transmissão original, exigindo uma administração cuidadosa para controlar quais bytes foram recebidos corretamente. Porém, cada byte no fluxo tem seu próprio deslocamento exclusivo, isso pode ser feito.

O TCP deve estar preparado para lidar com todos esses problemas e resolvê-los de maneira eficiente. Foi feito um grande esforço no sentido de otimizar o desempenho dos fluxos TCP, mesmo diante dos problemas da rede. A seguir, descreveremos diversos algoritmos usados por muitas implementações do TCP.

[T3] 6.5.4 O cabeçalho de segmento do TCP

A Figura 6.29 mostra o layout de um segmento TCP. Cada segmento começa com um cabeçalho de formato fixo de 20 bytes. O cabeçalho fixo pode ser seguido por opções de cabeçalho. Depois das opções, se for o caso, pode haver até 65.535 – 20 – 20 = 65.495 bytes de dados, onde o primeiro valor 20 se refere ao cabeçalho IP e o segundo ao cabeçalho TCP. Segmentos sem quaisquer dados são válidos e são comumente usados para confirmações e mensagens de controle.

[arte: ver original p. 537]

[Dísticos]

[1] 32 bits

[2]Source port Destination port

Sequence number

Acknowledgement number

[3]TCP header length URG ACK PSH PST SYN FIN Window size

Checksum Urgent pointer

[4] Options (0 ou mais palavras de 32 bits)

Dados (campo opcional)

[F]Figura 6.29

[FL] O cabeçalho TCP

Analisaremos o cabeçalho TCP campo a campo. Os campos *Source port* e *Destination port* identificam os pontos terminais locais da conexão. As portas conhecidas são definidas em *www.iana.org*, mas cada host pode alocar as outras portas como desejar. Uma porta e o endereço IP de seu host formam um único ponto terminal de 48 bits. Os números dos pontos terminais de origem e de destino identificam a conexão.

Os campos *Sequence number* e *Acknowledgement number* desempenham suas funções habituais. Observe que o segundo especifica o próximo byte esperado e não o último byte recebido corretamente. Ambos têm 32 bits, pois cada byte de dados é numerado em um fluxo TCP.

O campo *TCP header length* informa quantas palavras de 32 bits existem no cabeçalho TCP. Essa informação é necessária, porque o campo *Options* tem tamanho variável; assim, o mesmo acontece com o cabeçalho. Tecnicamente, na verdade, esse campo indica o início dos dados dentro do segmento com base em palavras de 32 bits, mas esse número é apenas o tamanho do cabeçalho em

palavras e, portanto, o efeito é o mesmo.

Em seguida, temos um campo de 6 bits que não é utilizado. O fato desse campo ter sobrevivido intacto por mais de um quarto de século é a prova de como o TCP é bem organizado. Protocolos menores teriam precisado dele para corrigir bugs no projeto original.

Agora temos seis flags de 1 bit. O valor 1 é atribuído a *URG* se *Urgent pointer* estiver sendo usado. *Urgent pointer* é usado para indicar um deslocamento de bytes a partir do número de seqüência atual em que os dados urgentes devem ser encontrados. Esse recurso substitui as mensagens interrompidas. Como já mencionamos, esse recuso representa uma forma estruturada de permitir que o transmissor envie um sinal ao receptor sem envolver o serviço TCP no motivo da interrupção.

Ao bit *ACK* é atribuído o valor 1 para indicar que *Acknowledgement number* é válido. Se *ACK* for igual a zero, isso significa que o segmento não contém uma confirmação e assim o campo *Acknowledgement number* é ignorado.

O bit *PSH* indica dados com o flag PUSH. Com ele, o receptor é solicitado a entregar os dados à aplicação mediante sua chegada, em vez de armazená-los até que um buffer completo tenha sido recebido (o que ele poderia fazer para manter a eficiência).

O bit *RST* é utilizado para reinicializar uma conexão que tenha ficado confusa devido a uma falha no host ou por qualquer outra razão. Ele também é utilizado para rejeitar um segmento inválido ou para recusar uma tentativa de conexão. Em geral, se receber um segmento com o bit *RST* ativado, isso significa que você tem um problema.

O bit *SYN* é usado para estabelecer conexões. A solicitação de conexão tem *SYN* = 1 e *ACK* = 0 para indicar que o campo de confirmação de piggyback não está sendo utilizado. A resposta contém uma confirmação e, portanto, tem *SYN* = 1 e

ACK = 1. Basicamente, o bit *SYN* é usado para denotar CONNECTION REQUEST e CONNECTION ACCEPTED, enquanto o bit *ACK* é usado para distinguir entre essas duas possibilidades.

O bit *FIN* é utilizado para encerrar uma conexão. Ele indica que o transmissor não tem mais dados a enviar. Entretanto, um processo pode continuar a receber dados indefinidamente, mesmo depois da conexão ter sido encerrada. Tanto o segmento *SYN* quanto o segmento *FIN* têm números de seqüência e, portanto, são processados na ordem correta.

O controle de fluxo no TCP é administrado por meio de uma janela deslizante de tamanho variável. O campo *Window size* indica quantos bytes podem ser enviados a partir do byte confirmado. Um campo *Window size* igual a 0 é válido e informa que todos os bytes até *Acknowledgement number* – 1 inclusive foram recebidos, mas que o receptor precisa de um descanso no momento e agradeceria muito se nenhum outro dado fosse enviado. A permissão para retomar a transmissão de dados pode ser enviada mais tarde com o mesmo *Acknowledgement number* e com um campo *Window size* diferente de zero.

Nos protocolos do Capítulo 3, as confirmações de quadros recebidos e a permissão para enviar novos quadros eram mantidas juntas. Isso era uma conseqüência do tamanho fixo da janela para cada protocolo. No TCP, as confirmações e a permissão para enviar dados adicionais são completamente isoladas. Na verdade, um receptor pode dizer: "Recebi os bytes até *k*, mas não quero mais agora". Esse desacoplamento (na verdade, uma janela de amanho variável) proporciona flexibilidade adicional. Vamos estudá-lo em detalhes a seguir.

O campo *Checksum* também é fornecido para aumentar a confiabilidade. Ele confere o total de verificação do cabeçalho, dos dados e do pseudocabeçalho ilustrado na Figura 6.30. Ao efetuar esse cálculo, o campo *Checksum* do TCP é

definido como zero, e o campo de dados é preenchido com um byte zero adicional, caso seu tamanho seja um número ímpar. O algoritmo de total de verificação simplesmente soma todas as palavras de 16 bits em complementos de 1 e depois tira o complemento de 1 da soma. Desse modo, quando o receptor efetuar o cálculo no segmento inteiro, incluindo o campo *Checksum*, o resultado deverá ser 0.

[arte: ver original p. 539]

[Dísticos]

[1] 32 bits

[2] Source address

[3] Destination address

[4] 0 0 0 0 0 0 0 0 Protocol = 6 TCP segment length

[F]Figura 6.30

[FL] O pseudocabeçalho incluído no total de verificação do TCP

O pseudocabeçalho contém os endereços IP de 32 bits das máquinas de origem e de destino, o número do protocolo para TCP (6) e a contagem de bytes para o segmento TCP (incluindo o cabeçalho). Incluir o pseudocabeçalho no cálculo do total de verificação ajuda a detectar pacotes extraviados; no entanto, essa estratégia viola a hierarquia do protocolo, pois os endereços IP nele contidos deveriam estar na camada IP e não na camada TCP. O UDP utiliza o mesmo pseudocabeçalho em seu total de verificação.

O campo *Options* foi projetado como uma forma de oferecer recursos extras, ou seja, recursos que não foram previstos pelo cabeçalho comum. A opção mais importante é aquela que permite a cada host estipular o máximo de carga útil do TCP que está disposto a receber. O uso de segmentos grandes é mais eficiente do que a utilização de segmentos pequenos, pois o cabeçalho de 20 bytes pode ser

diluído em um maior volume de dados; porém, é possível que hosts pequenos não sejam capazes de administrar segmentos muito grandes. Durante a configuração da conexão, cada lado pode anunciar sua capacidade máxima e avaliar a capacidade de seu parceiro. Se um host não usar essa opção, o valor padrão de 536 bytes será estipulado para a carga útil. Todos os hosts da Internet são obrigados a aceitar segmentos TCP de 536 + 20 = 556 bytes. O tamanho máximo de segmento nos dois sentidos não precisa ser o mesmo.

Para linhas com alta largura de banda, alto retardo ou ambos a janela de 64 KB é quase sempre um problema. Em uma linha T3 (44,736 Mbps), são necessários apenas 12 ms para enviar uma janela de 64 KB completa. Se o retardo da propagação da viagem de ida e volta for de 50 ms (o mais comum em um cabo de fibra óptica transcontinental), o transmissor ficará inativo durante 3/4 do tempo, aguardando confirmações. Em uma conexão por satélite, a situação é ainda pior. Um tamanho de janela maior permitiria que o transmissor continuasse enviando os dados; no entanto, com um campo *Window size* de 16 bits, não é possível expressar esse tamanho. Na RFC 1323 foi proposta uma opção *Window scale*, permitindo ao transmissor e ao receptor negociar um fator de escala para a janela. Esse número permite a ambos os lados da transmissão deslocar o campo *Window size* até 14 bits para a esquerda, gerando assim janelas de até 2^{30} bytes. A maior parte das implementações do TCP já é compatível com essa opção.

Outra opção proposta pela RFC 1106 e agora amplamente implementada é o uso da retransmissão seletiva, em vez do protocolo go back n. Se um receptor receber um segmento defeituoso seguido de um grande número de segmentos perfeitos, o protocolo TCP normal eventualmente sofrerá um timeout e retransmitirá todos os segmentos não confirmados, incluindo os que chegaram em perfeitas condições (isto é, o protocolo go back n). A RFC 1106 introduziu o uso de NAKs para permitir que o receptor solicite um segmento (ou segmentos) específico(s).

Depois de receber esses segmentos, o receptor poderá confirmar todos os dados armazenados no buffer, reduzindo assim o volume de dados a serem retransmitidos.

[T3] 6.5.5 O estabelecimento de conexões TCP

As conexões são estabelecidas no TCP por meio do handshake de três vias discutido na Seção 6.2.2. Para estabelecer uma conexão, um lado — digamos, o servidor — aguarda passivamente por uma conexão de entrada, executando as primitivas LISTEN e ACCEPT através da especificação de uma determinada origem ou de ninguém em particular.

O outro lado — digamos, o cliente — executa uma primitiva CONNECT, especificando o endereço IP e a porta à qual deseja se conectar, o tamanho máximo do segmento TCP que está disposto a aceitar, opcionalmente, alguns dados do usuário (por exemplo, uma senha). A primitiva CONNECT envia um segmento TCP com o bit *SYN* ativado e um bit *ACK* desativado, e aguarda uma resposta.

Quando esse segmento chega ao destino, a entidade TCP dessa estação verifica se existe um processo que tenha executado uma primitiva LISTEN na porta informada no campo *Destination port*. Caso contrário, ela envia uma resposta com o bit *RST* ativado para rejeitar a conexão.

Se algum processo estiver na escuta da porta, esse processo receberá o segmento TCP de entrada. Em seguida, ele poderá aceitar ou rejeitar a conexão. Se o processo aceitar, um segmento de confirmação será retornado. A seqüência dos segmentos TCP enviados em condições normais é ilustrada na Figura 6.31(a). Observe que um segmento *SYN* consome 1 byte de espaço de seqüência, para que seja confirmado sem ambigüidade.

No caso de dois hosts tentarem estabelecer uma conexão entre os mesmo

soquetes simultaneamente, ocorrerá a seqüência de eventos ilustrada na Figura 6.31(b). O resultado desses eventos é o estabelecimento de apenas uma conexão e não de duas, porque as conexões são identificadas por suas extremidades. Se a primeira configuração resultar em uma conexão identificada por (*x*, *y*) e a segunda também, haverá somente uma entrada na tabela para (*x*, *y*).

[arte: ver original p. 540]

[Dísticos]

[1] Tempo

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 6.31

[FL] (a) Estabelecimento de uma conexão TCP em condições normais. (b) Colisão entre chamadas

O número de seqüência inicial de uma conexão não é 0, pelas razões que discutimos antes. É utilizado um esquema baseado no uso de um clock, com um pulso de clock a cada 4 μs. Como proteção adicional, quando um host sofrer uma pane, ele não poderá se reinicializar por um período equivalente à duração máxima de pacotes, a fim de ter certeza de que não haverá pacotes de conexões anteriores ainda presentes em algum lugar na Internet.

Apesar das conexões TCP serem full-duplex, fica mais fácil compreender como as conexões são encerradas se as considerarmos um par de conexões simplex. Cada conexão simplex é encerrada de modo independente de sua parceira. Para encerrar uma conexão, qualquer dos lados pode enviar um segmento com o bit *FIN* ativado, o que significa que não há mais dados a serem transmitidos. Quando *FIN* é confirmado, esse sentido é desativado para novos dados. No entanto, os

dados podem continuar a fluir indefinidamente no outro sentido. Quando os dois sentidos da conexão estiverem desativados, a conexão será encerrada. De modo geral, são necessários quatro segmentos TCP para encerrar uma conexão, isto é, um *FIN* e um *ACK* para cada sentido. Porém, é possível que o primeiro ACK e o segundo *FIN* ocupem o mesmo segmento, o que baixa o número total para três. A exemplo do que ocorre com as ligações telefônicas em que duas pessoas dizem adeus e desligam simultaneamente, é possível que as duas extremidades da conexão TCP enviem segmentos *FIN* ao mesmo tempo. Eles são confirmados do modo habitual, e a conexão é desativada. Na verdade, não há qualquer diferença essencial entre as situações em que os dois hosts encerram a conexão de forma seqüencial ou simultânea.

Para evitar que ocorra o problema dos dois exércitos, são utilizados timers. Se uma resposta para um *FIN* não chegar no período equivalente a duas vezes o tempo máximo de duração de um pacote, o transmissor do *FIN* encerrará a conexão. Eventualmente, o outro lado perceberá que não há mais ninguém na escuta e também sofrerá um timeout. Mesmo que essa solução não seja perfeita, pois a solução perfeita é teoricamente impossível, ela terá de bastar. Na prática, os problemas são raros.

[T3] 6.5.7 Modelagem do gerenciamento de conexões do TCP

As etapas necessárias para o estabelecimento e o encerramento de conexões podem ser representadas em uma máquina de estados finitos com os 11 estados mostrados na Figura 6.32. Em cada estado, determinados eventos são válidos. Quando ocorre um evento válido, é possível executar uma ação. Se ocorrer algum outro evento, será reportado um erro.

Cada conexão começa no estado *CLOSED*. Ela sai desse estado ao executar uma abertura passiva (LISTEN) ou ativa (CONNECT). Se o outro lado executar a

primitiva oposta, a conexão será estabelecida e o estado passará a ser *ESTABLISHED*. O encerramento da conexão pode ser iniciado por qualquer um dos lados. Quando ele se completa, o estado volta a ser *CLOSED*.

A máquina de estados finitos propriamente dita está ilustrada na Figura 6.33. Uma situação comum em que um cliente se conecta ativamente a um servidor passivo é representado pelas linhas mais escuras — a linha contínua para o cliente e a linha tracejada para o servidor. As linhas mais claras representam seqüências de eventos incomuns. Cada linha na Figura 6.33 é marcada por um par *evento/ação*. O evento pode ser uma chamada de sistema iniciada pelo usuário (CONNECT, LISTEN, SEND ou CLOSE), uma chegada de segmento (*SYN*, *FIN*, *ACK* ou *RST*) ou, em um único caso, um período de timeout igual a duas vezes a duração máxima dos pacotes. A ação é o envio de um segmento de controle (*SYN*, *FIN* ou *RST*) ou nada, indicado por um travessão (—). Os comentários são mostrados entre parênteses.

[arte: ver original p. 542]

[T]Tabela

Estado	**Descrição**
CLOSED	Nenhuma conexão está ativa ou pendente
LISTEN	O servidor está esperando pela chegada de uma chamada
SYN RCVD	Uma solicitação de conexão chegou; espera por ACK
SYN SENT	A aplicação começou a abrir uma conexão
ESTABLISHED	O estado normal para a transferência de dados
FIN WAIT 1	A aplicação informou que acabou de transmitir
FIN WAIT 2	O outro lado concordou em encerrar
TIMED WAIT	Aguarda a entrega de todos os pacotes
CLOSING	Ambos os lados tentaram encerrar a transmissão simultaneamente
CLOSE WAIT	O outro lado deu início a uma encerramento

LAST ACK Aguarda a entrega de todos os pacotes

[F]Figura 6.32

[FL] Os estados usados na máquina de estados finitos para o gerenciamento de uma conexão TCP

O diagrama pode ser melhor compreendido se seguirmos primeiro o caminho de um cliente (a linha escura contínua) e depois o caminho do servidor (a linha escura tracejada). Quando uma aplicação na máquina cliente emite uma solicitação CONNECT, a entidade TCP local cria um registro de conexão, assinala que a conexão se encontra no estado *SYN SENT* e envia um segmento *SYN*. Observe que muitas conexões podem estar abertas (ou sendo abertas) ao mesmo tempo por várias aplicações; portanto, o estado se refere a cada conexão e é incluído no registro de conexões. Quando *SYN* + *ACK* chega, o TCP envia o *ACK* final do handshake de três vias e passa para o estado *ESTABLISHED*. Só então os dados podem ser transmitidos e recebidos.

Quando uma aplicação é encerrada, ela executa uma primitiva CLOSE, o que faz com que a entidade TCP local envie um segmento *FIN* e aguarde o *ACK* correspondente (o quadro tracejado marca o fechamento ativo). Quando *ACK* chega, há uma transição para o estado *FIN WAIT* 2 e um sentido da conexão é desativado. Quando o outro lado também for desativado, chegará um *FIN* que será confirmado. Agora os dois lados estão desativados, mas o TCP aguarda um período equivalente ao tempo máximo de duração de um pacote para ter certeza de que todos os pacotes da conexão foram recebidos, caso alguma confirmação tenha se perdido. Quando o timer expirar, o TCP removerá o registro da conexão.

Examinaremos agora o gerenciamento de conexões do ponto de vista do servidor. O servidor executa uma primitiva LISTEN e aguarda para ver quem aparece. Quando um *SYN* chegar, ele será confirmado e o servidor passará para o estado

SYN RCVD. Quando o *SYN* do servidor for confirmado, o handshake de três vias ficará completo e o servidor passará para o estado *ESTABLISHED*. Nesse caso, a transferência de dados já pode ocorrer.

[arte: ver original p. 543]

[Dísticos]

[1] (Início)

CONNECT/SYN (Etapa 1 do handshake de três vias)

CLOSED

CLOSE/-

LISTEN/- **CLOSE**/-

SYN/SYN + ACK

LISTEN

(Etapa 2 do handshake de três vias)

[2]SYN RCVD RST/- **SEND**/SYN SYN SENT

SYN/SYN + ACK (abertura simultânea)

[3](Estado de transferência de dados)

ACK/- ESTABLISHED SYN + ACK/ACK

(Etapa 3 do handshake de três vias)

CLOSE/FIN

[4] **CLOSE**/FIN FIN/ACK

(Fechamento ativo) (Fechamento passivo)

[5] FIN WAIT 1 FIN/ACK CLOSING

ACK/- ACK/-

FIN + ACK/ACK

FIN WAIT 2 FIN/ACK TIME WAIT

(Timeout/)

CLOSED ACK/-

(Voltar ao início)

[6] CLOSE WAIT

CLOSE/FIN

LAST ACK

[F]Figura 6.33

[FL] Máquina de estados finitos usada no gerenciamento de uma conexão TCP. A linha contínua mais escura representa o caminho normal de um cliente. A linha tracejada mais escura representa o caminho normal de um servidor. As linhas mais finas representam eventos incomuns. Cada transição é identificada pelo evento que a provoca e pela ação resultante dela, separados por uma barra

Quando estiver satisfeito, o cliente executará uma primitiva CLOSE, o que faz com que um *FIN* chegue ao servidor (o quadro tracejado marca o fechamento passivo). Em seguida, o servidor recebe um sinal. Quando ele também executar uma primitiva GLOSE, um *FIN* será enviado ao cliente. Quando a confirmação do cliente for recebida, o servidor encerrará a conexão e apagará o registro da conexão.

[T3] 6.5.8 Política de transmissão do TCP

Como mencionamos antes, o gerenciamento de janelas no TCP não está diretamente vinculado às confirmações, como acontece na maioria dos protocolos de enlace de dados. Por exemplo, suponha que o receptor tenha um buffer de 4096 bytes, como ilustra a Figura 6.34. Se o transmissor enviar um segmento de 2048 bytes e este for recebido de forma correta, o receptor confirmará o segmento. Porém, como agora ele só tem 2048 bytes de espaço disponível em seu buffer (até que alguma aplicação retire alguns dados do buffer), o receptor anunciará uma janela de 2048 bytes começando no próximo byte esperado.

[arte: ver original p. 544]

[Dísticos]

[1] Transmissor

A aplicação faz uma gravação de 2 KB

A aplicação faz uma gravação de 2 KB

O transmissor é bloqueado

O transmissor pode enviar até 2 KB

[2] Receptor Buffer do receptor

0 4 KB

Vazio

2 KB

Cheio

A aplicação lê 2 KB

2 KB

1 KB 2 KB

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 6.34

[FL] Gerenciamento de janelas no TCP

Agora o transmissor envia outros 2048 bytes, que são confirmados, mas a janela anunciada é 0. O transmissor deve parar até o processo de aplicação no host receptor remover alguns dados do buffer, quando então o TCP poderá anunciar uma janela maior.

Quando a janela é 0, o transmissor não pode enviar segmentos da forma como faria sob condições normais, mas há duas exceções. Na primeira, os dados urgentes podem ser enviados para, por exemplo, permitir que o usuário elimine o

processo executado na máquina remota. Na segunda, o transmissor pode enviar um segmento de 1 byte para fazer com que o receptor anuncie novamente o próximo byte esperado e o tamanho da janela. O padrão TCP oferece essa opção de forma explícita para evitar um impasse no caso de um anúncio de janela se perder.

Os transmissores não são obrigados a enviar os dados assim que os recebem das aplicações. Nem os receptores têm a obrigação de enviar as confirmações imediatamente. Por exemplo, na Figura 6.34, quando os primeiros 2 KB de dados chegaram, o TCP, sabendo que havia uma janela de 4 KB disponível, estaria completamente correto se apenas armazenasse os dados no buffer até chegarem outros 2 KB, a fim de poder transmitir um segmento com 4 KB de carga útil. Essa liberdade pode ser explorada para melhorar o desempenho das conexões.

Considere uma conexão telnet com um editor interativo que reage a cada tecla pressionada. Na pior das hipóteses, quando um caractere chegar à entidade TCP receptora, o TCP cria um segmento TCP de 21 bytes, que será repassado ao IP para ser enviado como um datagrama IP de 41 bytes. No lado receptor, o TCP envia imediatamente uma confirmação de 40 bytes (20 bytes de cabeçalho TCP e 20 bytes de cabeçalho IP). Mais tarde, quando o editor tiver lido o byte, o TCP enviará uma atualização de janela, movendo a janela um byte para a direita. Esse pacote também tem 40 bytes. Por último, quando o editor tiver processado o caractere, ele o ecoará como um pacote de 41 bytes. No total, 162 bytes de largura de banda são utilizados e quatro segmentos são enviados para cada caractere digitado. Quando a largura de banda é escassa, esse método não se mostra uma boa opção.

Uma abordagem usada por muitas implementações do TCP para otimizar essa situação é retardar as confirmações e atualizações de janelas durante 500 ms, na esperança de encontrar algum dado que lhes dê "uma carona". Supondo que o

editor ecoe dentro de 500 ms, apenas um pacote de 41 bytes precisará ser retornado ao usuário remoto, reduzindo à metade a contagem de pacotes e o uso da largura de banda.

Embora essa regra reduza a carga imposta à rede pelo receptor, o transmissor ainda estará operando de modo ineficiente, enviando pacotes de 41 bytes que contêm apenas um byte de dados. Uma forma de reduzir esse uso é conhecida como **algoritmo de Nagle** (Nagle, 1984). A sugestão de Nagle é simples: quando os dados chegarem ao transmissor um byte por vez, basta enviar o primeiro byte e armazenar no buffer todos os outros, até que o byte pendente tenha sido confirmado. Em seguida, envie todos os caracteres armazenados no buffer em um único segmento TCP e comece a armazenar no buffer novamente, até todos os caracteres terem sido confirmados. Se o usuário estiver digitando com rapidez e a rede for lenta, cada segmento poderá conter um número significativo de caracteres, reduzindo bastante a largura de banda utilizada. O algoritmo permite ainda que um novo pacote seja enviado se houver dados suficientes para preencher metade da janela ou um segmento máximo.

O algoritmo de Nagle é amplamente utilizado por implementações do TCP, mas há ocasiões em que é melhor desativá-lo. Em particular, quando uma aplicação X Windows está sendo executada na Internet, os movimentos do mouse devem ser enviados ao computador remoto. (O sistema X Windows é o sistema de janelas utilizado na maioria dos sistemas UNIX.) Agrupá-los para depois enviá-los em rajadas faz com que o cursor do mouse se movimente de forma errática, o que deixa os usuários insatisfeitos.

Outro problema que pode arruinar o desempenho do TCP é a **síndrome da janela boba** (Clark, 1982). Esse problema ocorre quando os dados são repassados para a entidade TCP transmissora em grandes blocos, mas uma aplicação interativa no lado receptor lê os dados um byte por vez. Para entender o problema, observe a

Figura 6.35. Inicialmente, o buffer TCP no lado receptor está cheio e o transmissor sabe disso (ou seja, ele tem uma janela de tamanho 0). Em seguida, uma aplicação interativa lê um caractere do fluxo TCP. Essa ação faz com que o TCP receptor fique satisfeito e envie uma atualização de janela ao transmissor, informando que ele pode enviar 1 byte. O transmissor agradece e envia 1 byte. Agora, o buffer se enche outra vez; portanto, o receptor confirma o segmento de 1 byte e atribui o valor 0 ao tamanho da janela. Esse comportamento pode durar para sempre.

[arte: ver original p. 546]

[Dísticos]

[1] Cabeçalho

Cabeçalho

1 Byte

[2] O buffer do receptor está cheio

A aplicação lê 1 byte

Espaço para mais um byte

Segmento de atualização de janela enviado

Chegou um novo byte

O buffer do receptor está cheio

[F]Figura 6.35

[FL] Síndrome da janela boba

A solução apresentada por Clark é evitar que o receptor envie uma atualização de janela para 1 byte. Em vez disso, ele é forçado a aguardar até que haja um espaço considerável na janela para então anunciar o fato. Para ser mais específico, o receptor não deve enviar uma atualização de janela até que possa lidar com o tamanho máximo de segmento que anunciou quando a conexão foi estabelecida,

ou até que seu buffer esteja com metade de sua capacidade livre (o que for menor).

Além disso, o transmissor também pode ajudar não enviando segmentos muito pequenos. Ele deve tentar aguardar até ter acumulado espaço suficiente na janela para enviar um segmento inteiro ou pelo menos um segmento que contenha dados equivalentes à metade da capacidade do buffer no lado receptor (o que ele deve estimar a partir do padrão de atualizações de janelas que já recebeu).

O algoritmo de Nagle e a solução de Clark para a síndrome da janela boba são complementares. Nagle tentava resolver o problema causado pelo fato de a aplicação transmissora entregar dados ao TCP um byte por vez. Já Clark tentava acabar com o problema criado pelo fato de a aplicação receptora retirar os dados do TCP um byte por vez. Ambas as soluções são válidas e podem funcionar juntas. O objetivo é evitar que o transmissor envie segmentos pequenos e que o receptor tenha de solicitá-los.

O TCP receptor pode fazer mais para melhorar o desempenho da rede do que apenas realizar a atualização de janelas em unidades grandes. Assim como o TCP transmissor, ele também tem a capacidade de armazenar dados; portanto, pode bloquear uma solicitação READ da aplicação até ter um bom volume de dados a oferecer. Isso reduz o número de chamadas ao TCP e, conseqüentemente, o overhead. É óbvio que isso também aumenta o tempo de resposta; no entanto, para aplicações não interativas como a transferência de arquivos, a eficiência prevalece sobre o tempo de resposta a solicitações individuais.

Outro problema para o receptor é o que fazer com os segmentos fora de ordem. Eles podem ser mantidos ou descartados, a critério do receptor. Obviamente, as confirmações só podem ser enviadas quando todos os dados até o byte confirmado tiverem sido recebidos. Se o receptor receber os segmentos 0, 1, 2, 4, 5, 6 e 7, ele poderá confirmar tudo até o último byte do segmento 2, inclusive.

Quando o transmissor sofrer um timeout, ele retransmitirá o segmento 3. Se tiver armazenado os segmentos de 4 a 7, o receptor poderá, ao receber o segmento 3, confirmar todos os bytes até o fim do segmento 7.

[T3] 6.5.9 Controle de congestionamento do TCP

Quando a carga oferecida a qualquer rede é maior que sua capacidade, acontece um congestionamento. A Internet não é exceção a essa regra. Nesta seção, descreveremos algoritmos que foram desenvolvidos no último quarto de século para lidar com o congestionamento. Embora a camada de rede também tente gerenciar o congestionamento, o trabalho mais pesado é feito pelo TCP, pois a verdadeira solução para o congestionamento é diminuir a taxa de transmissão de dados.

Na teoria, é possível lidar com o congestionamento aplicando-se um princípio emprestado da física: a lei da conservação de pacotes. A idéia é não injetar um novo pacote na rede até que um pacote antigo sai da rede (ou seja, até que o pacote antigo tenha sido entregue). O TCP tenta alcançar esse objetivo manipulando dinamicamente o tamanho da janela.

O primeiro passo para gerenciar o congestionamento é detectá-lo. Antigamente, detectar um congestionamento era difícil. Um timeout causado por um pacote perdido podia ter sido provocado por (1) ruído em uma linha de transmissão ou (2) pelo fato de o pacote ter sido descartado em um roteador congestionado. Era difícil saber a diferença entre os dois casos.

Hoje em dia, a perda de pacotes devido a erros de transmissão é relativamente rara, porque a maioria dos troncos de longa distância é de fibra (embora também existam as redes sem fios). Como conseqüência, a maioria dos timeouts de transmissão na Internet se deve a congestionamentos. Todos os algoritmos TCP da Internet pressupõem que os timeouts são causados por congestionamentos, e

monitoram os timeouts à procura de sinais de dificuldades, da mesma forma que os trabalhadores das minas de carvão observam o comportamento de seus canários para verificar se há formação de gases venenosos.

Antes de discutirmos como o TCP reage aos congestionamentos, vamos descrever o que ele faz para tentar evitar sua ocorrência. Quando uma conexão é estabelecida, deve-se escolher um tamanho de janela adequado. O receptor pode especificar uma janela a partir do tamanho de seu buffer. Se o transmissor se mantiver dentro do tamanho da janela, não haverá problemas causados pela sobrecarga dos buffers na extremidade receptora, mas eles ainda poderão ocorrer devido a congestionamentos internos na rede.

Na Figura 6.36, vemos esse problema ilustrado por um exemplo hidráulico. Na Figura 6.36(a), vemos um cano que leva a um recipiente de pequena capacidade. Desde que o transmissor não envie mais água do que o balde pode conter, não haverá perda de água. Na Figura 6.36(b), o fator limitador não é a capacidade do balde, mas sim a capacidade interna da rede. Se entrar água demais na rede com muito rapidez, haverá retenção e uma parte da água será perdida (nesse caso, o funil transbordará).

A solução da Internet é entender que existem dois problemas potenciais — a capacidade da rede e a capacidade do receptor — e lidar com cada um deles separadamente. Para isso, cada transmissor mantém duas janelas: a janela fornecida pelo receptor e uma segunda janela, a **janela de congestionamento.** Cada uma reflete o número de bytes que o transmissor pode enviar. O número de bytes que podem ser transmitidos é o valor mínimo entre as duas janelas. Desse modo, a janela efetiva é a janela mínima que o transmissor imagina ser correta, e que o receptor também imagina ser correta. Se o receptor pedir "Envie 8 KB", mas o transmissor souber que qualquer rajada com mais de 4 KB irá congestionar a rede, ele enviará apenas 4 KB. Por outro lado, se o receptor pedir "Envie 8 KB", e o

transmissor souber que rajadas de até 32 KB passam pela rede sem problemas, ele enviará os 8 KB solicitados.

[arte: ver original p. 548]

[Dísticos]

[1] Ajuste de taxa de transmissão

[2] Rede de transmissão

[3] Receptor de pequena capacidade

(a)

[4] Congestionamento interno

[5] Receptor de grande capacidade

(b)

[F]Figura 6.36

[FL] (a) Uma rede rápida alimentando um receptor de pequena capacidade. (b) Uma rede lenta alimentando um receptor de grande capacidade

Quando uma conexão é estabelecida, o transmissor ajusta a janela de congestionamento ao tamanho do segmento máximo em uso na conexão. Em seguida, ele envia um segmento máximo. Se esse segmento for confirmado antes de ocorrer um timeout, o transmissor incluirá o número de bytes de um segmento na janela de congestionamento, de modo que ela tenha capacidade equivalente a dois segmentos máximos, e em seguida enviará dois segmentos. À medida que cada um desses segmentos for confirmado, a janela de congestionamento será aumentada em um tamanho de segmento máximo. Quando a janela de congestionamento chegar a *n* segmentos, se todos os *n* segmentos forem confirmados a tempo, a janela de congestionamento será aumentada no número de bytes correspondente a *n* segmentos. Na prática, cada rajada confirmada duplica a janela de congestionamento.

A janela de congestionamento mantém seu crescimento exponencial até que ocorra um timeout ou que a janela do receptor seja alcançada. Se rajadas de tamanho igual a 1024, 2048 e 4096 bytes funcionarem bem, mas uma rajada de 8192 bytes causar um timeout, a janela de congestionamento deverá ser mantida em 4096 bytes para evitar congestionamentos. Desde que a janela de congestionamento seja mantida com 4096 bytes, nenhuma rajada superior a esse tamanho será enviada, não importando quanto espaço de janela o receptor ofereça. Esse algoritmo é conhecido como **inicialização lenta**, mas ele não é lento de modo algum (Jacobson, 1988). Esse algoritmo é exponencial. Todas as implementações do TCP devem ser compatíveis com ele.

Agora, vamos ver o algoritmo de controle de congestionamento da Internet. Ele utiliza um terceiro parâmetro, o **limiar**, inicialmente igual a 64 KB, além das janelas do receptor e de congestionamento. Quando há um timeout, o limiar é definido como a metade da janela de congestionamento atual, e a janela de congestionamento é redefinida como um segmento máximo. Em seguida, a inicialização lenta é usada para determinar o que a rede é capaz de gerenciar, exceto pelo fato de o crescimento exponencial ser interrompido quando o limiar é alcançado. A partir daí, as transmissões bem-sucedidas proporcionam um crescimento linear à janela de congestionamento (o aumento é de um segmento máximo para cada rajada) em vez de um para cada segmento. Na prática, esse algoritmo parte do princípio de que deve ser aceitável reduzir à metade a janela de congestionamento, e depois retomar seu crescimento a partir daí.

O funcionamento desse algoritmo é ilustrado na Figura 6.37. O tamanho máximo do segmento aqui é 1024 bytes. Em princípio, a janela de congestionamento tinha 64 KB, mas ocorreu um timeout e assim o limiar foi definido como 32 KB, e a janela de congestionamento foi definida como 1 KB para a transmissão 0. Em seguida, a janela de congestionamento cresce de exponencialmente até chegar ao

limiar (32 KB). A partir daí, seu crescimento é linear.

A transmissão 13 não tem muita sorte (ela já deveria saber disso), e ocorre um timeout. Ao limiar é atribuído um valor igual à metade da janela atual (agora, 40 KB e, portanto, a metade da janela corresponde a 20 KB), e o início lento recomeça. Quando as confirmações da transmissão 14 começam a chegar, cada uma das quatro primeiras duplica a janela de congestionamento mas, depois disso, o crescimento volta a ser linear.

Se não ocorrerem outros timeouts, a janela de congestionamento continuará a crescer até atingir o tamanho da janela do receptor. Nesse ponto, ela pára de crescer e permanece constante desde que não ocorra outro timeout, e desde que a janela do receptor não mude de tamanho. Por outro lado, se um pacote SOURCE QUENCH do ICMP chegar e for repassado ao TCP, esse evento será tratado como se tivesse ocorrido um timeout. Uma abordagem alternativa (e mais recente) é descrita na RFC 3168.

[arte: ver original p. 550]

[Dísticos]

[1] Janela de congestionamento (kilobytes)

[2] Timeout

[3] Limiar

[4] Limiar

[5] Número da transmissão

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 6.37

[FL] Um exemplo de algoritmo de congestionamento da Internet

[T3] 6.5.10 Gerenciamento de timers do TCP

O TCP utiliza vários timers (pelo menos conceitualmente) para realizar seu trabalho. O mais importante deles é o **timer de retransmissão.** Quando um segmento é enviado, um timer de retransmissão é ativado. Se o segmento for confirmado antes do timer expirar, ele será interrompido. Por outro lado, se o timer expirar antes da confirmação chegar, o segmento será retransmitido (e o timer disparado mais uma vez). Com isso, surge a seguinte pergunta: Qual deve ser o intervalo de timeout?

Esse problema é mais difícil na camada de transporte da Internet do que nos protocolos de enlace de dados genéricos do Capítulo 3. Neles, o retardo esperado é bastante previsível (isto é, há uma pequena variância); portanto, o timer pode ser programado para expirar logo após o momento em que a confirmação é esperada, como mostra a Figura 6.38(a). Como raramente as confirmações se atrasam na camada de enlace de dados (por não haver congestionamentos), a ausência de uma confirmação no momento esperado em geral significa que o quadro ou a confirmação se perdeu.

[arte: ver original p. 551]

[Dísticos]

[1] Probabilidade

Tempo do percurso de ida e volta (ms)

(a)

[2] Probabilidade

Tempo do percurso de ida e volta (ms)

(b)

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 6.38

[FL] (a)Densidade de probabilidades de tempos de chegada de confirmações na camada de enlace de dados. (b) Densidade de probabilidades de tempos de chegada de confirmações para o TCP

O TCP encontra um ambiente radicalmente distinto. A função da densidade de probabilidades para o tempo que uma confirmação TCP leva para retornar é mais semelhante à Figura 6.38(b) do que à Figura 6.38(a). Determinar o tempo de ida e volta para o destino não é uma tarefa fácil. Mesmo quando o tempo é conhecido, também é difícil decidir sobre o intervalo de timeout. Se o timeout for curto demais, digamos T_1 na Figura 6.38(b), ocorrerão retransmissões desnecessárias, sobrecarregando a Internet com pacotes inúteis. Por outro lado, se ele for longo demais (por exemplo, T_2), o desempenho será prejudicado devido ao longo retardo de retransmissão sempre que um pacote se perder. Além disso, a média e a variância na distribuição da chegada das confirmações podem mudar com muita rapidez em um intervalo de poucos segundos, devido à ocorrência ou à resolução de um congestionamento.

A solução é utilizar um algoritmo altamente dinâmico que ajuste constantemente os intervalos de timeout, com base na contínua avaliação do desempenho da rede. O algoritmo quase sempre utilizado pelo TCP foi criado por Jacobson (1988) e funciona da seguinte forma: para cada conexão, o TCP mantém uma variável, *RTT*, que é a melhor estimativa no momento para o tempo de percurso de ida e volta até o destino em questão. Quando um segmento é enviado, um timer é disparado, não só para verificar quanto tempo a confirmação leva para chegar, como também para acionar uma retransmissão, caso a confirmação demore demais. Se a confirmação voltar antes do timer expirar, o TCP medirá o tempo necessário, que será *M*. Em seguida, o TCP atualiza a variável *RTT* de acordo com

a fórmula:

$RTT = \alpha RTT + (1 - \alpha)M$

onde α é um fator de suavização que determina o peso dado ao antigo valor. Normalmente α = 7/8.

Mesmo que o valor para *RTT* seja adequado, a escolha de um timeout de retransmissão apropriado não é uma tarefa fácil. Em geral, o TCP utiliza β*RTT* mas, nesse caso, a dificuldade é determinar o valor de β. Nas implementações iniciais, β era sempre 2, mas a experiência mostrou que um valor constante era inflexível, porque não atendia aos casos em que a variância era maior.

Em 1988, Jacobson propôs tornar β aproximadamente proporcional ao desvio padrão da função de densidade de probabilidades de tempos de chegada de confirmações; assim, uma grande variância significa um alto valor de β e vice-versa. Em particular, ele sugeriu o uso do *desvio médio* como uma forma econômica de estimar o *desvio padrão*. O algoritmo de Jacobson exige o controle de outra variável de suavização, *D*, o desvio. Sempre que uma confirmação chega, é calculada a diferença entre os valores esperados e os valores observados, $|RTT - M|$. O valor suavizado dessa expressão é mantido em *D* pela fórmula:

[Inserir equação do O.A. p. 552a]

onde α pode ter ou não o mesmo valor usado para suavizar *RTT*. Mesmo que a variável *D* não seja exatamente igual ao desvio padrão, ela atende às necessidades da rede. Jacobson também demonstrou que essa variável pode ser calculada usando-se apenas somas, subtrações e deslocamento de inteiros, o que é uma grande vantagem. A maioria das implementações do TCP atuais utiliza esse algoritmo e define o intervalo de timeout como:

[Inserir equação do O.A. p. 552b]

A escolha do fator 4 é um tanto arbitrária, mas tem duas vantagens. A primeira é que as multiplicações por 4 podem ser realizadas com um único deslocamento. E

a segunda é que esse fator minimiza o número de timeouts e retransmissões desnecessárias, porque menos de 1% de todos os pacotes chega com um atraso de mais de quatro vezes o desvio padrão. (Na verdade, em princípio, Jacobson sugeriu que o fator usado fosse 2, mas o trabalho com as redes mostrou que 4 oferece um desempenho melhor.)

Um problema com a estimativa dinâmica de *RTT* é o que fazer quando um segmento sofre um timeout e é retransmitido. Quando uma confirmação chega, não fica claro se ela se refere à primeira transmissão ou a uma transmissão posterior. Uma suposição errada pode comprometer seriamente a estimativa de *RTT*. Phil Karn descobriu esse problema da maneira mais difícil. Ele é um entusiasta do radioamadorismo, e interessa muito pela transmissão de pacotes TCP/IP por rádio, um meio notoriamente não confiável (em um bom dia, apenas metade dos pacotes chega a seu destino). Ele fez uma proposta simples: não atualizar *RTT* em qualquer segmento que tenha sido retransmitido. Em vez disso, o intervalo de timeout seria duplicado a cada falha, até os segmentos chegarem ao destino da primeira vez. Essa correção é conhecida como **algoritmo de Karn.** A maioria das implementações do TCP utiliza esse algoritmo.

O timer de retransmissão não é o único timer que o TCP utiliza. Há outro timer, chamado **timer de persistência.** Ele foi projetado para evitar o impasse descrito a seguir. O receptor envia uma confirmação com um tamanho de janela 0, pedindo ao transmissor para esperar. Mais tarde, o receptor atualiza a janela, mas o pacote com a atualização se perde. Agora, tanto o transmissor quanto o receptor estão aguardando que o outro faça alguma coisa. Quando o timer de persistência expirar, o transmissor enviará um teste ao receptor. A resposta ao teste fornece o tamanho da janela. Se ela ainda for zero, o timer de persistência será ativado novamente e o ciclo se repetirá. Se for diferente de zero, os dados poderão ser transmitidos.

Um terceiro timer utilizado por algumas implementações é o **timer keepalive** (manter vivo). Quando uma conexão permanece inativa por muito tempo, o timer keepalive pode expirar para fazer um lado verificar se o outro lado ainda está ativo. Se esse outro lado não responder, a conexão será encerrada. Esse recurso é polêmico porque, além de aumentar o overhead, pode encerrar uma boa conexão devido a uma partição transitória na rede.

O último timer usado em cada conexão TCP é o timer empregado no estado *TIMED WAIT* durante o encerramento. Ele é executado por um tempo igual a duas vezes a duração máxima dos pacotes para garantir que, quando uma conexão for fechada, todos os pacotes criados por ela serão extintos.

[T3] 6.5.11 TCP e UDP sem fios

Na teoria, os protocolos de transporte deveriam ser independentes da tecnologia da camada de rede em que se baseiam. Particularmente, o TCP não deveria se preocupar com o fato do IP estar sendo executado sobre fibra ou rádio. Na prática, isso é importante, pois a maioria das implementações do TCP foi otimizada com todo o cuidado, de acordo com suposições que são verdadeiras para as redes fisicamente conectadas, mas que falham no caso das redes sem fios. Ignorar as propriedades da transmissão sem fio pode levar a uma implementação do TCP logicamente correta, mas que tem um desempenho catastrófico.

O principal problema é o algoritmo de controle de congestionamento. Quase todas as implementações do TCP atuais pressupõem que os timeouts ocorrem devido a congestionamentos, e não a pacotes perdidos. Conseqüentemente, quando um timer expira, o TCP diminui o ritmo e começa a transmitir de modo mais lento (por exemplo, o algoritmo de início lento de Jacobson). A idéia por trás dessa abordagem é reduzir a carga da rede e assim diminuir o congestionamento.

No entanto, infelizmente, os enlaces de dados das transmissões sem fios não são confiáveis. Eles perdem pacotes o tempo todo. A melhor estratégia para lidar com pacotes perdidos é enviá-los novamente o mais rápido possível. Diminuir o ritmo nesse caso tornará a situação ainda pior. Se, digamos, 20% de todos os pacotes se perderem, quando o transmissor enviar 100 pacotes/segundo, o throughput será de 80 pacotes/segundo. Se o transmissor diminuir a carga para 50 pacotes/segundo, o throughput cairá para 40 pacotes/segundo.

Em uma rede fisicamente conectada, quando um pacote é perdido, o transmissor deve diminuir o ritmo. Quando isso ocorre em uma rede sem fio, o transmissor deve aumentar ainda mais o ritmo. Se o transmissor não souber em que tipo de rede está trabalhando, será difícil tomar a decisão correta.

Com freqüência, o caminho entre o transmissor e o receptor não é homogêneo. Os primeiros 1000 km podem ser controlados por uma rede fisicamente conectada, enquanto o último 1 km pode representar uma rede sem fio. Nessas circunstâncias, é mais difícil ainda tomar uma decisão em relação ao timeout, pois é necessário saber onde ocorreu o problema. A solução proposta por Bakne e Badrinath (1995), denominada **TCP indireto**, consiste em dividir a conexão TCP em duas conexões separadas, como ilustra a Figura 6.39. A primeira conexão vai do transmissor à estação base. A segunda vai da estação base ao receptor. A estação base simplesmente copia pacotes entre as conexões em ambos os sentidos.

A vantagem desse mecanismo é que agora as duas conexões são homogêneas. Os timeouts da primeira conexão podem fazer com que o transmissor diminua o ritmo, enquanto os da segunda podem torná-la mais rápida. Outros parâmetros também podem ser ajustados separadamente para as duas conexões. A desvantagem é que o mecanismo viola a semântica do TCP. Como cada parte da conexão é uma conexão TCP completa, a estação base confirma cada segmento

TCP da maneira usual. A diferença é que, nesse caso, quando o transmissor recebe uma confirmação, isso não quer dizer que o segmento chegou ao receptor, mas sim à estação base.

Uma outra solução, criada por Balakrishnan *et al*. (1995), não rompe a semântica do TCP. Ela faz várias modificações pequenas no código da camada de rede da estação base. Uma das mudanças é a inclusão de um espião que observa e armazena em um cache os segmentos TCP que saem para o host móvel e as confirmações que retornam desse host. Quando o espião vê um segmento TCP seguir para o host móvel, mas não vê uma confirmação retornar antes de seu timer expirar (um tempo relativamente curto), ele apenas retransmite esse segmento sem informar o que está fazendo à origem. O espião também retransmite quando vê confirmações duplicadas do host móvel, o que invariavelmente quer dizer que o host móvel perdeu algo. As confirmações duplicadas são imediatamente descartadas para evitar que a origem as interprete de modo incorreto como um sinal de congestionamento.

[arte: ver original p. 554]

[Dísticos]

[1] Transmissor

[2] TCP #1

[3] Estação base

[4] TCP #2

[5] Host móvel

[6] Antena

[7] Roteador

[F]Figura 6.39

[FL] A divisão de uma conexão TCP em duas conexões

Porém, uma desvantagem dessa transparência é que, se o enlace de dados sem fio perder muitos pacotes, a máquina de origem poderá sofrer um timeout enquanto espera por uma confirmação e invocar o algoritmo de controle de congestionamento. Com o TCP indireto, o algoritmo de controle de congestionamento jamais será inicializado, a menos que exista de fato um congestionamento na parte fisicamente conectada da rede.

O ensaio de Balakrishnan *et al*. também apresenta uma solução para o problema dos segmentos perdidos originados no host móvel. Quando a estação base percebe um intervalo nos números de seqüência de entrada, ela solicita uma retransmissão seletiva dos bytes perdidos usando uma opção do TCP.

Essas soluções tornam o enlace de dados sem fio mais confiável em ambos os sentidos, sem que a estação de origem saiba disso e sem alterar a semântica do TCP.

Apesar de o UDP não ter os mesmos problemas do TCP, a comunicação sem fio também cria algumas dificuldades para ele. O principal problema é que os programas utilizam o UDP esperando que ele seja altamente confiável. Esses programas sabem que não há nenhuma garantia, mas mesmo assim esperam que ele seja quase perfeito. Em um ambiente sem fios, o UDP estará longe de ser perfeito. Para programas que conseguem se recuperar de mensagens UDP perdidas apenas a um custo considerável, a passagem repentina de um ambiente no qual as mensagens teoricamente podem se perder, mas raras vezes se perdem, para um ambiente em que a perda de mensagens é constante pode ter um efeito catastrófico sobre o desempenho.

A comunicação sem fio também afeta outras áreas, e não apenas o desempenho. Por exemplo, de que maneira um host móvel encontra uma impressora local para se conectar, em vez de usar sua impressora original? De alguma forma, isso está relacionado ao modo de se obter a página da Web que corresponde à célula local,

mesmo que seu nome não seja conhecido. Além disso, os projetistas de páginas da WWW tendem a supor que sempre haverá muita largura de banda disponível. No entanto, o uso de um logotipo muito grande em todas as páginas será contraproducente, se forem necessários 10 segundos para transmitir o logotipo por um enlace sem fio muito lento toda vez que houver uma referência à página, pois isso irritará os usuários.

À medida que as redes sem fios se tornam mais comuns, os problemas na execução do TCP sobre essas redes ficam mais agudos. Outras experiências e outros trabalhos nessa área são relatados em (Barakat *et al.*, 2000; Ghani e Dixit, 1999; Huston, 2001; e Xylomenos *et al.*, 2001).

[T3] 6.5.12 TCP transacional

Em uma seção anterior deste capítulo, estudamos a chamada de procedimentos remotos como um caminho para implementar sistemas cliente/servidor. Se a solicitação e a resposta forem pequenas o suficiente para se ajustarem em pacotes únicos e a operação for idempotente, será possível empregar apenas o UDP. Porém, se essas condições não forem satisfeitas, o uso do UDP será menos interessante. Por exemplo, se a resposta puder ser bastante grande, então os fragmentos deverão ser organizados em seqüência, e será preciso criar um mecanismo para retransmitir fragmentos perdidos. Na realidade, a aplicação será obrigada a reinventar o TCP.

É claro que isso não é nem um pouco interessante, mas usar o próprio TCP também não é interessante. O problema é a eficiência. A seqüência normal de pacotes para fazer uma RPC sobre o TCP é mostrada na Figura 6.40(a). Nove pacotes são exigidos no melhor caso.

Os nove pacotes são:

1. O cliente envia um pacote *SYN* para estabelecer uma conexão.

2. O servidor envia um pacote *ACK* para confirmar o pacote *SYN*.

3. O cliente completa o handshake de três vias.

4. O cliente envia a solicitação real.

5. O cliente envia um pacote *FIN* para indicar que concluiu a transmissão.

6. O servidor confirma a solicitação e o *FIN*.

7. O servidor devolve a resposta ao cliente.

8. O servidor envia um pacote *FIN* para indicar que também terminou.

9. O cliente confirma o pacote *FIN* do servidor.

Observe que esse é o melhor caso. No pior caso, a solicitação do cliente e o pacote *FIN* são conformados separadamente, da mesma forma que a resposta do servidor e o pacote *FIN*.

[arte: ver original p. 556]

[Dísticos]

[1] Tempo

[2] Cliente Servidor

[3] solicitação

[4] solicitação + FIN

[5] resposta

[6] Tempo

[7] Cliente Servidor

[8] solicitação

[9] resposta

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 6.40

[FL] (a) RPC usando o TCP normal. (b) RPC usando o T/TCP

A pergunta que surge de imediato é se existe algum modo de combinar a eficiência da RPC usando o UDP (apenas duas mensagens) com a confiabilidade do TCP. A resposta é: Quase. Isso pode ser feito com uma variante experimental do TCP chamada **T/TCP (Transactional TCP)**, descrita nas RFCs 1379 e 1644.

A idéia central aqui é modificar ligeiramente a seqüência normal de configuração da conexão, a fim de permitir a transferência de dados durante a configuração. O protocolo T/TCP é ilustrado na Figura 6.40(b). O primeiro pacote do cliente contém o bit *SYN*, a solicitação propriamente dita e o *FIN*. Na prática, isso significa: "Quero estabelecer uma conexão, aqui estão os dados e terminei".

Quando o servidor recebe a solicitação, ele pesquisa ou calcula a resposta, e escolhe como responder. Se a resposta couber em um pacote, ele transmitirá a resposta da Figura 6.40(b), que diz: "Confirmo seu *FIN*, aqui está a resposta e terminei. O cliente então confirma o *FIN* do servidor, e o protocolo termina com três mensagens.

Porém, se o resultado for maior que um pacote, o servidor também terá a opção de não ativar o bit *FIN*, e nesse caso poderá enviar vários pacotes antes de fechar a conexão em seu sentido.

Vale a pena mencionar que o T/TCP não é o único aperfeiçoamento proposto para TCP. Outra proposta é o **SCTP (Stream Control Transmission Protocol).** Suas características incluem: preservação dos limites de mensagens, vários modos de entrega (por exemplo, entrega não ordenada), multihoming (destinos de reserva) e confirmações seletivas (Stewart e Metz, 2001). Porém, toda vez que alguém propõe mudar algo que funcionou tão bem por tanto tempo, existe sempre uma enorme batalha entre os grupos que defendem os lemas: "Os usuários estão exigindo mais recursos" e "Em time que está ganhando não se mexe".

[T2] 6.6 Questões de desempenho

As questões referentes ao desempenho são muito importantes nas redes de computadores. Quando centenas de milhares de computadores estão interconectados, são comuns interações complexas que trazem conseqüências imprevistas. Com freqüência, essa complexidade resulta em um fraco desempenho, cujas razões todos desconhecem. Nas próximas seções, examinaremos várias questões relacionadas ao desempenho das redes, a fim de constatarmos que tipos de problemas existem e o que pode ser feito para solucioná-los.

Infelizmente, compreender o desempenho de uma rede é mais uma arte que uma ciência. Pouco do que existe em termos de teoria é realmente útil na prática. O que podemos fazer é oferecer regras práticas que aprendemos com a experiência e apresentar exemplos reais. Deixamos essa discussão para depois do estudo da camada de transporte nas redes TCP de propósito, para podermos usar o TCP como exemplo.

A camada de transporte não é o único elemento em que surgem questões de desempenho. Vimos algumas delas na camada de rede, no capítulo anterior. Mesmo assim, a camada de rede tende a estar relacionada com o roteamento e o controle de congestionamento. Em geral, as questões mais amplas e orientadas para o sistema tendem a se relacionar ao transporte; assim, este capítulo é um local apropriado para examiná-las.

Nas cinco seções seguintes, estudaremos cinco aspectos do desempenho das redes:

1. Problemas de desempenho.
2. Medição do desempenho da rede.
3. Projeto de um sistema para proporcionar um desempenho melhor.
4. Processamento rápido de TPDUs.

5. Protocolos para as futuras redes de alto desempenho.

Por outro lado, precisamos de um nome genérico para designar as unidades intercambiadas pelas entidades de transporte. O termo usado no TCP, segmento, é no mínimo confuso e nunca foi usado fora do TCP nesse contexto. Os termos empregados em redes ATM (CS-PDU, SAR-PDU e CPCS-PDU) são específicos do ATM. O termo pacote se refere claramente à camada de rede, enquanto mensagem se refere à camada de aplicação. Na falta de um termo padrão, voltaremos a chamar de TPDUs as unidades intercambiadas pelas entidades de transporte. Quando quisermos fazer referência à TPDU e ao pacote juntos, utilizaremos pacote para identificar o conjunto, como em: "A CPU deve ser rápida o suficiente para processar os pacotes recebidos em tempo real". Aqui, nos referimos tanto ao pacote da camada de rede quanto à TPDU encapsulada nesse pacote.

[T3] 6.6.1 Problemas de desempenho em redes de computadores

Alguns problemas de desempenho, como o congestionamento, são causados pela sobrecarga temporária de recursos. Se, de repente, um roteador receber um tráfego maior do que é capaz de manipular, haverá um congestionamento e uma queda de desempenho. Estudamos o congestionamento em detalhes no capítulo anterior.

O desempenho também é prejudicado quando há um desequilíbrio nos recursos estruturais. Por exemplo, se uma linha de comunicação de gigabits estiver associada a um PC com poucos recursos, a fraca CPU não será capaz de processar os pacotes recebidos com a rapidez necessária, e alguns deles serão perdidos. Esses pacotes serão retransmitidos, aumentando o retardo, desperdiçando largura de banda e reduzindo o desempenho.

As sobrecargas também podem ser disparadas de forma sincronizada. Por

exemplo, se uma TPDU contiver um parâmetro inadequado (como a porta a que ela se destina), em muitos casos o receptor, muito solícito, retornará uma notificação de erro. Agora, imagine o que poderia acontecer se uma TPDU inadequada fosse transmitida por difusão para 10.000 máquinas; cada uma delas poderia enviar de volta uma mensagem de erro. Isso causaria um congestionamento de transmissões (ou tempestade de transmissões) que poderia paralisar a rede. O UDP sofreu com esse problema até ser alterado para impedir que os hosts respondessem a erros nas TPDUs do UDP enviadas a endereços de difusão.

Um segundo exemplo de sobrecarga sincronizada é o que acontece depois de uma falha no fornecimento de energia elétrica. Quando a energia retorna, todas as máquinas acessam suas ROMs ao mesmo tempo para acionar a rotina de reinicialização. Uma seqüência de reinicialização típica poderia exigir o acesso a um servidor qualquer (DHCP), com a finalidade de confirmar a identidade do usuário, e depois a algum servidor de arquivos para obter uma cópia do sistema operacional. Se centenas de máquinas fizerem isso ao mesmo tempo, o servidor com certeza interromperá seu funcionamento devido à sobrecarga.

Mesmo na ausência de sobrecargas sincronizadas e quando há recursos suficientes disponíveis, pode ocorrer um desempenho fraco devido à falta de ajuste do sistema. Por exemplo, em uma máquina com uma CPU potente e muita memória, mas com pouca memória alocada para o espaço de buffers, haverá sobrecargas, e muitas TPDUs serão perdidas. Da mesma forma, se o algoritmo de programação de execução não der prioridade suficiente ao processamento das TPDUs recebidas, algumas delas poderão se perder.

Outra questão relacionada ao ajuste é definir os timeouts de forma correta. Quando uma TPDU é enviada, em geral um timer é ativado para evitar sua perda. Se o intervalo de temporização for muito curto, haverá retransmissões

desnecessárias, aumentando o volume do tráfego. Se, ao contrário, o intervalo for muito longo, ocorrerão retardos desnecessários após a perda de uma TPDU.
Outros parâmetros ajustáveis incluem o tempo de espera por dados para enviar uma confirmação por piggyback antes de decidir enviar uma confirmação separada, e ainda o número de retransmissões antes de uma desistência.
As redes de gigabits trazem com elas novos problemas de desempenho. Por exemplo, considere a hipótese de transmitir uma rajada de dados de 64 KB de San Diego a Boston, a fim de preencher o buffer de 64 KB do receptor. Suponha que o enlace tenha 1 Gbps e que o retardo unidirecional da velocidade da luz na fibra seja 20 ms. Inicialmente, em $t = 0$, o canal está vazio, como ilustra a Figura 6.41(a). Apenas 500 µs depois, na Figura 6.41(b), todas as TPDUs estão no cabo de fibra. A primeira TPDU estará em algum lugar nas vizinhanças de Brawley, ainda no sul da Califórnia. Entretanto, o transmissor deverá parar até receber uma atualização de janela.

[arte: ver original p. 559]

[Dísticos]

[1] Dados

(a) (b)

[2] Confirmações

(c) (d)

[F]Figura 6.41

[FL] O estado de transmissão de um megabit de San Diego a Boston. (a) No instante $t = 0$. (b) Após 500 µs. (c) Após 20 ms. (d) Após 40 ms

Depois de 20 ms, a primeira TPDU chega a Boston, como mostra a Figura 6.41(c), e é confirmada. Por fim, 40 ms depois do início do procedimento, a primeira confirmação chega ao transmissor e a segunda rajada pode ser transmitida.

Como a linha de transmissão foi usada durante apenas 0,5 ms em 40 ms, a eficiência é de 1,25%. Essa é uma situação típica da execução de protocolos antigos sobre linhas de gigabits.

Uma quantidade útil que se deve ter em mente ao se analisar o desempenho das redes é o **produto da largura de banda pelo retardo**. Esse produto é obtido multiplicando-se a largura de banda (em bits/segundo) pelo tempo de retardo da viagem de ida e volta (em segundos). O produto é a capacidade do canal desde o transmissor até o receptor e de volta (em bits).

No exemplo da Figura 6.41, o produto da largura de banda pelo retardo é igual a 40 milhões de bits. Em outras palavras, o transmissor teria de enviar uma rajada de 40 milhões de bits para manter a velocidade máxima até a chegada da primeira confirmação. São necessários todos esses bits para encher o canal (em ambos os sentidos). É por isso que uma rajada de meio milhão de bits só alcança uma eficiência de 1,25%: ela equivale a apenas 1,25% da capacidade do canal.

A conclusão a que podemos chegar é que, para se obter um bom desempenho, a janela do receptor tem de ser pelo menos tão grande quanto o produto da largura de banda pelo retardo, de preferência um pouco maior, pois talvez o receptor não responda no mesmo instante. Para uma linha de gigabits transcontinental, são exigidos pelo menos 5 megabytes.

Se a eficiência é terrivelmente baixa para a transmissão de um megabit, imagine como ela seria para atender a uma solicitação de algumas centenas de bytes. A menos que se possa descobrir outro uso para a linha enquanto o primeiro cliente está aguardando a resposta, uma linha de gigabits não é melhor do que uma linha de megabits; ela só é mais cara.

Outro problema de desempenho que ocorre com as aplicações em que o tempo de transmissão tem importância fundamental, como sinais de áudio e vídeo, é a flutuação. Não basta ter um tempo de transmissão curto, também é necessário

um pequeno desvio padrão. Porém, conseguir um tempo médio de transmissão pequeno juntamente com um desvio padrão pequeno requer um grande esforço de engenharia.

[T3] 6.6.2 Medição do desempenho da rede

Quando uma rede tem baixo desempenho, em geral os usuários reclamam com seus administradores, exigindo melhorias. Para melhorar o desempenho, os operadores devem primeiro descobrir exatamente o que está acontecendo. Para isso, os operadores precisarão fazer medições. Nesta seção, veremos as medições do desempenho da rede. O estudo a seguir se baseia no trabalho de Mogul (1993).

O loop básico usado para melhorar o desempenho da rede inclui as seguintes etapas:

1. Medir os parâmetros relevantes e o desempenho da rede.
2. Tentar entender o que está acontecendo.
3. Alterar um parâmetro.

Essas etapas são repetidas até que o desempenho seja bom o suficiente ou até que esteja claro que foi feito todo o possível para melhorar o desempenho.

As medições podem ser feitas de várias maneiras e em diferentes pontos da rede (tanto nos elementos físicos quanto na pilha de protocolos). O tipo mais elementar de medição consiste em ativar um timer ao se iniciar um procedimento e usá-lo com a finalidade de verificar o tempo necessário para concluir essa atividade. Por exemplo, é fundamental saber quanto tempo é necessário para uma TPDU ser confirmada. Outras medições são realizadas com contadores que registram a freqüência com que algum evento aconteceu (por exemplo, o número de TPDUs perdidas). Por último, sempre há um interesse em saber a quantidade de algo, como o número de bytes processados em um certo intervalo de tempo.

A medição dos parâmetros e do desempenho das redes tem muitas armadilhas potenciais. Apresentaremos a seguir uma lista com algumas delas. Qualquer tentativa sistemática de medir o desempenho das redes deve ter o cuidado de evitar essas armadilhas.

[T4] Certifique-se de que o tamanho da amostra é grande o bastante

Não meça o tempo necessário para enviar uma TPDU, mas repita a medição, digamos, um milhão de vezes e tire a média. Com uma grande amostra, será possível reduzir a incerteza (a variação) na medição da média e do desvio padrão. Essa incerteza pode ser calculada com o emprego de fórmulas estatísticas convencionais.

[T4] Certifique-se de que as amostras são representativas

O ideal é que toda a seqüência de um milhão de medições seja repetida em dias e horários distintos para que o efeito de diferentes cargas do sistema sobre a quantidade medida possa ser verificado. Por exemplo, medições de congestionamento terão pouca utilidade se forem feitas nos momentos em que não há congestionamento. Algumas vezes, talvez os resultados não pareçam intuitivos a princípio, como no caso de um grande congestionamento que acontece às 10, 11, 13 e 14 horas, mas não ao meio-dia (quando todos os usuários saem para almoçar).

[T4] Tenha cuidado ao usar o clock do computador

Os clocks dos computadores funcionam incrementando algum contador a intervalos regulares. Por exemplo, um timer de milissegundos acrescenta uma unidade a um contador a cada milissegundo. A utilização de um timer para medir um evento que dura menos de 1 milissegundo não é impossível, mas requer um

certo cuidado. (É claro que alguns computadores têm clocks mais precisos.) Por exemplo, ao se medir o tempo necessário para enviar uma TPDU, o clock do sistema (digamos, em milissegundos) deve ser lido na entrada do código da camada de transporte e na saída desse código. Se o tempo de envio real de uma TPDU for de 300 µs, o valor da diferença entre as duas leituras será 0 ou 1, ambos errados. Entretanto, se a medição for repetida um milhão de vezes e o total das medições for somado e dividido por um milhão, o tempo médio terá uma precisão melhor que 1 µs.

[T4] Certifique-se de que nenhum evento inesperado está ocorrendo durante os testes

Se fizer medições no sistema de uma universidade no dia em que um grande projeto de laboratório tiver de ser entregue, você poderá obter resultados diferentes daqueles que seriam encontrados se a medição fosse feita no dia seguinte. De modo semelhante, se um pesquisador decidir fazer uma videoconferência pela rede enquanto você estiver realizando seus testes, os resultados obtidos poderão ser bem diferentes da realidade. A melhor opção é fazer os testes em um sistema inativo e criar você mesmo toda a carga de trabalho necessária. Até mesmo essa estratégia apresenta armadilhas. Embora você possa imaginar que ninguém estará usando a rede às 3 horas da manhã, esse pode ser exatamente o momento em que o programa de backup automático começa a copiar o conteúdo de todos os discos para uma fita. Além disso, talvez haja um tráfego intenso para as páginas da World Wide Web de que você mais gosta, devido às diferenças de fuso horário.

[T4] O uso de caches pode arruinar as medições

A forma mais óbvia de medir os tempos de transferência de arquivos é abrir um

arquivo extenso, ler seu conteúdo, fechar o arquivo e ver quanto tempo tudo isso demora. Depois, repita a medição muitas outras vezes para obter uma boa média. O problema é que o sistema pode armazenar o arquivo em um cache; nesse caso, somente a primeira medição envolveria realmente o tráfego da rede. Todas as outras medições só incluiriam a leitura do cache local. O resultado dessas medições é inútil (a menos que você queira avaliar o desempenho do cache).

E possível burlar o cache simplesmente excedendo sua capacidade. Por exemplo, se o cache tiver a capacidade de 10 MB, o loop de teste poderia abrir, ler e fechar dois arquivos de 10 MB em cada passagem, em uma tentativa de fazer a taxa de acesso ao cache chegar a 0. Mesmo assim, é necessário muito cuidado, a menos que você tenha certeza absoluta de que compreende o algoritmo de armazenamento no cache.

O armazenamento em buffer pode causar um efeito semelhante. Um conhecido programa utilitário de desempenho do TCP/IP ficou famoso por relatar que o UDP pode alcançar um desempenho substancialmente mais alto que o permitido pela linha física. Como isso acontece? Uma chamada ao UDP normalmente retorna um controle assim que a mensagem é aceita pelo núcleo (kernel) e acrescentada à fila de transmissão. Se houver espaço em buffer suficiente, sincronizar 1000 chamadas UDP não quer dizer que os dados foram enviados. A maioria deles ainda pode estar no núcleo, mas o utilitário de medição do desempenho imagina que todos foram transmitidos.

[T4] Entenda o que está medindo

Quando você mede o tempo necessário para ler um arquivo remoto, suas medições dependem da rede, dos sistemas operacionais do cliente e do servidor, das placas de interface de hardware utilizadas, de seus drivers e de outros fatores. Se as medições forem feitas com cuidado, você descobrirá o tempo de

transferência de arquivos para a configuração que está usando. Se o seu objetivo for ajustar essa configuração, essas medições serão de grande utilidade. Entretanto, se estiver realizando medições semelhantes em três sistemas distintos para escolher qual placa de interface de rede deve ser comprada, talvez os resultados sejam anulados pelo fato de um dos drivers de rede ser muito ruim e só utilizar 10% do potencial de desempenho da placa.

[T4] Tenha cuidado para não extrapolar os resultados
Suponha que você tenha realizado medições utilizando cargas de rede simuladas que variam de 0 (inativa) a 0,4 (40% de sua capacidade), como mostram os pontos de dados e a linha continua que passa por eles na Figura 6.42. Talvez seja interessante extrapolar os resultados linearmente, como mostra a linha pontilhada. Entretanto, muitos resultados de enfileiramento envolvem um fator $1/(1 - \rho)$, onde ρ é a carga; portanto, os valores verdadeiros devem ser mais parecidos com os valores da linha tracejada, que aumentam com rapidez muito maior que a linear.

[T3] 6.6.3 Projeto de sistemas para obter melhor desempenho
Medições e ajustes podem melhorar o desempenho consideravelmente, mas não podem substituir um bom projeto. Uma rede mal projetada pode ser melhorada, mas só até certo ponto. Daí em diante, ela precisa ser totalmente refeita.
Nesta seção, apresentaremos algumas regras práticas baseadas em nossa experiência com muitas redes. Essas regras se referem ao projeto de sistemas, e não apenas ao projeto de redes, pois o software e o sistema operacional muitas vezes são mais importantes que os roteadores e as placas de interface. Muitas dessas idéias são do conhecimento dos projetistas de redes há anos e têm sido passadas de geração a geração. Elas foram especificadas pela primeira vez por

Mogul (1993); nosso tratamento é paralelo ao dele. Outra fonte relevante é (Metcalfe, 1993).

[T4] Regra n° 1: A velocidade da CPU é mais importante que a velocidade da rede

Longos anos de experiência mostraram que em quase todas as redes, o sistema operacional e o overhead do protocolo dominam o tempo real no cabo. Por exemplo, teoricamente o tempo mínimo de RPC em uma rede Ethernet é 102 μs, o que corresponde a uma solicitação mínima (64 bytes) seguida por uma resposta mínima (64 bytes). Na prática, superar o tempo de overhead do software e fazer o tempo da RPC ficar próximo dele é um aperfeiçoamento substancial.

[arte: ver original p. 563]

[Dísticos]

[1] Tempo de resposta

[2] 5

4

3

2

1

0

[3] 0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0

[4] Carga

[F]Figura 6.42

[FL] Resposta como uma função da carga

Da mesma forma, o maior problema de se trabalhar a 1 Gbps é conseguir que os bits passem do buffer do usuário para o cabo de fibra com rapidez suficiente, e fazer com que a CPU receptora processá-los à mesma velocidade com que eles

chegam. Em resumo, se duplicar a velocidade da CPU, com freqüência você chegará perto de duplicar o throughput. Contudo, com freqüência duplicar a velocidade da rede não tem qualquer efeito prático, pois em geral o gargalo está nos hosts.

[T4]Regra nº 2: Reduza o número de pacotes para reduzir o overhead do software

O processamento de uma TPDU tem um certo volume de overhead para cada TPDU (por exemplo, o processamento do cabeçalho) e um certo volume de processamento por byte (por exemplo, calcular o total de verificação). Ao se transmitir 1 milhão de bytes, o overhead por byte é mesmo, não importando o tamanho da TPDU. Porém, usar TPDUs de 128 bytes significa criar um overhead 32 vezes maior por TPDU, em relação ao uso de TPDUs de 4 KB. Esse overhead cresce rapidamente.

Além do overhead da TPDU, deve-se considerar também o overhead nas camadas mais baixas. Cada pacote recebido causa uma interrupção. Em um processador moderno com pipeline, cada interrupção desfaz o pipeline da CPU, interfere com o cache, exige uma mudança no contexto de gerenciamento de memória e força a gravação de um número substancial de registradores da CPU. Uma redução de *n* vezes o número de TPDUs transmitidas causa uma redução de *n* vezes no overhead de interrupções e pacotes.

Essa observação sugere a coleta de um volume substancial de dados antes da transmissão, a fim de reduzir o número de interrupções na estação receptora. O algoritmo de Nagle e a solução de Clark para a síndrome da janela boba são tentativas de fazer exatamente isso.

[T4] Regra nº 3: Minimize as mudanças de contexto

As mudanças de contexto (por exemplo, do modo de núcleo para o modo de

usuário) são fatais. Elas têm as mesmas propriedades ruins que as interrupções, sendo a pior delas uma longa série de erros de cache iniciais. É possível reduzir as mudanças de contexto através de um procedimento de biblioteca que envia os dados para um buffer interno até que haja um volume substancial de dados. Da mesma forma, na estação receptora, as pequenas TPDUs recebidas devem ser agrupadas e repassadas ao usuário de uma só vez, em vez de serem repassadas individualmente, com a finalidade de minimizar as mudanças de contexto.

Na melhor das hipóteses, um pacote recebido provoca uma mudança de contexto do usuário atual para o núcleo, e depois uma mudança para o processo receptor, a fim de enviar os dados recém-chegados. No entanto, infelizmente em muitos sistemas operacionais acontecem outras mudanças. Por exemplo, se o gerenciador da rede executar um processo especial no espaço do usuário, a chegada de um pacote provavelmente causará uma mudança de contexto do usuário atual para o núcleo, depois uma outra mudança do núcleo para o gerenciador de rede, seguida por outra de volta ao núcleo, e uma última do núcleo para o processo receptor. Essa seqüência é ilustrada na Figura 6.43. Todas essas mudanças de contexto em cada pacote desperdiçam tempo de CPU e terão um efeito devastador sobre o desempenho da rede.

[arte: ver original p. 565]

[Dísticos]

[1] Processo do usuário em execução no momento da chegada do pacote

[2] Gerenciador de rede

[3] Processo receptor

[4] Espaço do usuário

[5] Espaço do núcleo

[6] 1 2 3 4

[F]Figura 6.43

[FL] Quatro mudanças de contexto para manipular um pacote por meio de um gerenciador de rede do espaço do usuário

[T4] Regra nº 4: Minimize o número de cópias

Pior que criar muitas mudanças de contexto é fazer muitas cópias. É comum um pacote ser copiado de três a quatro vezes antes da entrega da TPDU que ele contém. Depois de ser recebido pela interface de rede em um buffer de hardware especial, o pacote é copiado para um buffer do núcleo. Daí, ele é copiado para o buffer da camada de rede, depois para o buffer da camada de transporte e finalmente para o processo de aplicação receptor.

Um sistema operacional inteligente copiará uma palavra por vez, mas não é incomum precisar de até cinco instruções por palavra (carga, armazenamento, incremento de um registrador de índice, um teste para detectar o fim dos dados e um desvio condicional). A execução de três cópias de cada pacote em cinco instruções por palavra de 32 bits copiada exige 15/4 ou aproximadamente quatro instruções por byte copiado. Em uma CPU de 500 MIPS, uma instrução leva de 2 ns, e assim cada byte precisa de 8 ns de tempo de processamento, ou cerca de 1 ns por bit, o que corresponde a uma taxa máxima de quase 1 Gbps. Ao acrescentarmos o overhead para o processamento do cabeçalho, para o tratamento de interrupções e para mudanças de contexto, chegamos a uma taxa possível de 500 Mbps, e ainda nem sequer consideramos o processamento real dos dados. Fica claro que manipular uma rede Ethernet de 10 Gbps operando a plena velocidade está fora de questão.

De fato, até mesmo uma linha de 500 Mbps não pode ser manipulada à velocidade total. No cálculo anterior, supomos que uma máquina de 500 MIPS pode executar 500 milhões de instruções por segundo. Porém, na realidade, as máquinas só podem funcionar a essas velocidades se não tiverem de fazer

referência à memória. Com freqüência, as operações com a memória são dez vezes mais lentas que as instruções de registrador para registrador (isto é, 20 ns/instrução). Se 20% das instruções realmente fizerem referência à memória (ou seja, são erros de cache) o que é provável quando se trata de pacotes recebidos, o tempo médio de execução de instrução é 5,6 ns ($0,8 \times 2 + 0,2 \times 20$). Com quatro instruções/byte, precisamos de 22,4 ns/byte, ou 2,8 ns/ bit, o que dá cerca de 357 Mbps. Considerando um overhead de 50%, temos 178 Mbps. Observe que a assistência do hardware não ajudará muito nesse caso. O problema é o excesso de cópias pelo sistema operacional.

[T4] Regra nº 5: Você pode adquirir mais largura de banda, mas não um retardo mais baixo

As três regras a seguir lidam com a comunicação e não com o processamento de protocolos. A primeira regra estabelece que, se quiser mais largura de banda, você poderá simplesmente comprá-la. Colocar um segundo cabo de fibra ao lado do primeiro duplica a largura de banda, mas não altera em nada o retardo. Para tornar o retardo menor, é necessário melhorar o software do protocolo, o sistema operacional ou a interface de rede. E mesmo que tudo isso seja feito, o retardo não será reduzido se o gargalo for o tempo de transmissão.

[T4] Regra nº 6: Prevenir o congestionamento é melhor do que remediá-lo

A velha máxima de que é melhor prevenir do que remediar se aplica a essa situação. Quando uma rede está congestionada, pacotes são perdidos, há desperdício de largura de banda, retardos inúteis acontecem e muito mais. Remediar essa situação toma tempo e paciência. É muito melhor evitar que tudo isso aconteça. Impedir o congestionamento é como ser vacinado: dói um pouco quando você toma, mas evita algo que poderia causar dores muito maiores.

[T4] Regra nº 7: Evite os timeouts

Os timers são necessários nas redes, mas devem ser usados com moderação, e os timeouts devem ser minimizados. Quando um timer expira, em geral alguma ação é repetida. Se essa repetição for imprescindível, tudo bem, mas repeti-la sem necessidade é um desperdício.

O melhor modo de evitar trabalho desnecessário é cuidar para que os timers sejam programados de maneira um pouco conservadora. Um timer que demora tempo demais para expirar aumenta o retardo de uma conexão, caso haja a (improvável) perda de uma TPDU. Um timer que expira cedo demais reduz mais ainda o já escasso tempo da CPU, desperdiça largura de banda e aumenta a carga em dezenas de roteadores sem uma boa razão para tal.

[T3] 6.6.4 Processamento rápido de TPDUs

A moral da história que acabamos de mostrar é que o maior obstáculo para aumentar a velocidade das redes é o software dos protocolos. Nesta seção, veremos algumas maneiras de tornar esse protocolo mais rápido. Para obter mais informações, consulte (Clark *et al.*, 1989; e Chase *et al.*, 2001).

O overhead do processamento de TPDUs tem dois componentes: o overhead por TPDU e o overhead por byte. Ambos devem ser combatidos. O segredo para o processamento rápido de TPDUs é separar a situação normal (a transferência de dados em um sentido) e dar a ela um tratamento especial. Ainda que seja necessária uma seqüência especial de TPDUs para se chegar ao estado *ESTABLISHED*, uma vez nele, o processamento de TPDUs se dá de forma simples até um dos lados começar a fechar a conexão.

Vamos começar examinando o lado transmissor no estado *ESTABLISHED* quando há dados a serem transmitidos. Para facilitar a compreensão, faremos a suposição

de que a entidade de transporte está no núcleo, apesar das mesmas idéias também serem verdadeiras caso ela seja uma biblioteca ou um processo do espaço do usuário dentro do processo transmissor. Na Figura 6.44, o processo transmissor fica preso no núcleo para executar uma primitiva SEND. A primeira providência da entidade de transporte é realizar um teste para verificar se esse é o caso normal: o estado é *ESTABLISHED*, nenhum dos lados está tentando fechar a conexão, uma TPDU regular (e não fora da banda) completa está sendo transmitida e há espaço de janela suficiente no receptor. Se todas as condições forem atendidas, não serão necessários outros testes e será possível seguir o caminho mais rápido pela entidade de transporte transmissora. em geral, esse caminho é seguido durante a maior parte do tempo.

No caso normal, os cabeçalhos de TPDUs de dados consecutivas são quase idênticos. Para tirar proveito desse fato, um protótipo de cabeçalho é armazenado na entidade de transporte. No início do caminho rápido, ele é copiado com a maior velocidade possível para um buffer de rascunho, palavra por palavra. Esses campos que mudam de TPDU para TPDU são então sobregravados no buffer. Com freqüência, esses campos são derivados de variáveis de estado, como o próximo número de seqüência. Em seguida, um ponteiro para o cabeçalho da TPDU completo e um ponteiro para os dados do usuário são repassados à camada de rede. Aqui, pode-se utilizar a mesma estratégia (não mostrada na Figura 6.44). Por último, a camada de rede repassa o pacote resultante à camada de enlace de dados para transmissão.

Como um exemplo do funcionamento desse princípio na prática, consideraremos o TCP/IP. A Figura 6.45(a) mostra o cabeçalho do TCP. Os campos iguais entre TPDUs consecutivas em um fluxo de uma só via são mostrados sombreados. Tudo que a entidade de transporte transmissora tem de fazer é copiar as cinco palavras do protótipo do cabeçalho para o buffer de saída, fornecer o próximo número de

seqüência (copiando-o de uma palavra na memória), calcular o total de verificação e incrementar o número de seqüência na memória. Em seguida, a entidade pode passar o cabeçalho e os dados para um procedimento especial do IP, a fim de transmitir uma TPDU máxima regular. Depois disso, o IP copia seu protótipo de cabeçalho de cinco palavras [ver Figura 6.45(b)] para o buffer, preenche o campo *Identification* e calcula seu total de verificação. Com isso, o pacote fica pronto para transmissão.

[arte: ver original p. 567]

[Dísticos]

[1] S Processo transmissor

[2] Armadilha no núcleo para enviar TPDU

[3] Teste

[4] Processo receptor S

[5] TPDU repassada ao processo receptor

[6] Teste

[7] Rede

[F]Figura 6.44

[FL] O caminho rápido do transmissor para o receptor é ilustrado pela linha mais grossa. As etapas do processamento nesse caminho estão sombreadas

Agora, vamos ver o processamento do caminho rápido no lado receptor da Figura 6.44. A primeira etapa é localizar o registro de conexão para a TPDU recebida. No caso do TCP, o registro de conexão pode ser armazenado em uma tabela de hash cuja chave é alguma função simples nos dois endereços IP e das duas portas. Uma vez que o registro de conexão é localizado, ambos os endereços e ambas as portas devem ser comparados, a fim de se verificar se o registro correto foi encontrado.

[arte: ver original p. 568]

[Dísticos]

[1] Source port Destination port

Sequence number

Acknowledgement number

Len Unused Window size

Checksum Urgent pointer

(a)

[2] VER. IHL TOS Total length

Identification Fragment offset

TTL Protocol Header checksum Source address

Destination address

(b)

[F]Figura 6.45

[FL] (a) O cabeçalho TCP. (b) O cabeçalho IP. Em ambos os casos, os campos sombreados foram tirados do protótipo sem qualquer alteração

Uma otimização que freqüentemente acelera ainda mais a pesquisa de registros de conexões deve manter um ponteiro para a última conexão usada e experimentar esse registro primeiro. Clark *et al*. (1989) utilizaram essa opção e observaram uma taxa de aproveitamento de mais de 90%. Outras heurísticas de pesquisa são descritas em (McKenney e Dove, 1992).

Em seguida, a TPDU é analisada para verificar se é uma TPDU normal, ou seja, se o estado é *ESTABLISHED* , se nenhum dos lados está tentando encerrar a conexão, se a TPDU está completa e não tem flags especiais definidos, e se o número de seqüência é o esperado. Algumas instruções são necessárias para a realização desses testes. Se todas as condições forem atendidas, será chamado um

procedimento de caminho rápido especial do TCP.

O caminho rápido atualiza o registro da conexão e copia os dados para o usuário. Enquanto copia, ele calcula o total de verificação, eliminando assim uma passagem extra pelos dados. Se o resultado do total de verificação estiver correto, o registro da conexão será atualizado e uma confirmação será enviada de volta. O esquema geral de primeiro fazer uma verificação rápida para ver se o cabeçalho é o que se espera, e depois fazer um procedimento especial tratar esse caso é chamado **prognóstico de cabeçalho.** Muitas implementações do TCP o utilizam. Quando essa otimização e todas as outras descritas neste capítulo são aplicadas ao mesmo tempo, é possível fazer o TCP funcionar a 90% da velocidade de uma cópia local de memória para memória, supondo-se que a rede seja suficientemente rápida.

Duas outras áreas nas quais pode haver ganhos importantes de desempenho são o gerenciamento de buffers e o gerenciamento de timers. A grande questão no gerenciamento de buffers é evitar cópias desnecessárias, como mencionamos antes. O gerenciamento de timers é importante porque quase todos os timers programados não chegam a expirar. Eles são programados para prevenir a perda de TPDUs, mas a maioria das TPDUs chega de forma correta, assim como suas confirmações. Conseqüentemente, é importante otimizar o gerenciamento, no caso de timers que raras vezes expiram.

Um esquema comum é usar uma lista encadeada contendo os eventos de timers ordenados por hora de expiração. A entrada do cabeçalho contém um contador que informa quantos pulsos faltam para a expiração. Cada entrada sucessiva contém um contador que informa quantos pulsos a separa da anterior. Assim, se os timers expirarem em 3, 10 e 12 pulsos, os contadores serão 3, 7 e 2, respectivamente.

A cada pulso do relógio, o contador na entrada do cabeçalho é decrementado.

Quando ele chega a zero, seu evento é processado e o próximo item da lista torna-se o cabeçalho. Seu contador não precisa ser alterado. Nesse esquema, a inclusão e a exclusão de timers são operações de alto custo, com tempos de execução proporcionais ao tamanho da lista.

Uma estratégia mais eficiente pode ser usada se o intervalo máximo do timer for fixo e previamente conhecido. Aqui pode ser usado um array chamado **roda de sincronização**, como mostra a Figura 6.46. Cada slot corresponde a um pulso do clock. A hora atual é $T = 4$. Os timers foram programados para expirar a 3, 10 e 12 pulsos a partir de agora. Se, de repente, um novo timer for programado para expirar em sete pulsos, uma outra entrada será criada no slot 11. Da mesma forma, se o timer programado para $T + 10$ tiver de ser cancelado, a lista que começa no slot 14 deverá ser verificada, e a entrada correspondente removida. Observe que o array da Figura 6.46 não contém qualquer timer acima de $T + 15$.

[arte: ver original p. 569]

[Dísticos]

[1] Slot

0 Ponteiro para a lista de timers de T + 12

1 0

2 0

3 0

4 0 Hora atual, T

5 0

6 0

7 Ponteiro para a lista de timers de T + 3

8 0

9 0

10 0

11 0

12 0

13 0

14 Ponteiro para a lista de timers de T + 10

15 0

[F]Figura 6.46

[FL] Uma roda de sincronização

Quando há um pulso do clock, o ponteiro atual é avançado um slot (de forma circular). Em seguida, se a entrada indicada for diferente de zero, todos os seus timers serão processados. Muitas variações sobre essa idéia básica são descritas em (Varghese e Lauck, 1987).

[T3] 6.6.5 Protocolos para redes de gigabits

No começo da década de 1990, começaram a aparecer as redes de gigabits. A primeira reação das pessoas foi utilizar protocolos antigos nessas redes, mas logo surgiram vários problemas. Nesta seção, vamos descrever alguns desses problemas e as orientações que novos protocolos estão adotando para solucioná-los, à medida que migramos para redes cada vez mais rápidas.

O primeiro problema é que muitos protocolos utilizam números de seqüência de 32 bits. Quando a Internet começou, as linhas entre roteadores eram principalmente linhas dedicadas de 56 kbps, e assim um host funcionando à velocidade total levava mais de uma semana para circular pelos números de seqüência. Para os projetistas do TCP, 2^{32} era um valor bem próximo de infinito, porque havia pouco perigo de pacotes antigos ainda estarem em trânsito uma semana depois de terem sido transmitidos. Com a Ethernet de 10 Mbps, o tempo de repetição passou a ser de 57 minutos, muito mais curto, embora ainda

administrável. Com uma rede Ethernet de 1 Gbps despejando dados na Internet, são necessários cerca de 34 segundos para circular por todos os números de seqüência, um valor bem abaixo da duração máxima de um pacote na Internet (120 segundos). De repente, 2^{32} não é mais uma aproximação tão boa para infinito, pois um transmissor pode circular pelo espaço de seqüência enquanto ainda existem pacotes antigos. Porém, a RFC 1323 fornece uma válvula de escape.

O problema é que muitos projetistas de protocolos pressupõem, embora não o admitam realmente, que o tempo necessário para usar todo o espaço de números de seqüência é muito maior que o tempo máximo de duração de um pacote. Nesse caso, não haveria necessidade de se preocupar com o problema de ainda haver duplicatas antigas quando os números de seqüência começassem a se repetir. Em velocidades de gigabits, essa suposição não declarada cai por terra.

Um segundo problema é que a velocidade dos meios de comunicação tem melhorado com maior rapidez do que a velocidade dos computadores. (Observação para os engenheiros de computadores: arregacem as mangas e superem os engenheiros de comunicações! Contamos com vocês.) Na década de 1970, a ARPANET operava a uma velocidade de 56 kbps com computadores que funcionavam à velocidade de aproximadamente 1 MIPS. Os pacotes tinham 1008 bits; logo, a ARPANET tinha capacidade para transmitir cerca de 56 pacotes/segundo. Com quase 18 ms disponíveis para cada pacote, um host podia usar 18.000 instruções para processar um pacote. É claro que isso ocuparia toda a capacidade da CPU, mas ainda era possível usar 9000 instruções por pacote e ainda manter metade da CPU para realizar o trabalho pesado.

Compare esses números com os números dos modernos computadores de 1000 MIPS, que trocam pacotes de 1500 bytes sobre uma linha de gigabits. Agora, o fluxo de pacotes é de mais de 80.000 por segundo; portanto, o processamento

dos pacotes deve ser concluído em 6,25 µs, se quisermos reservar metade da capacidade da CPU para as aplicações. Em 6,25 µs, um computador de 1000 MIPS pode executar 6250 instruções, apenas 1/3 do que os hosts da ARPANET tinham disponível. Além disso, as modernas instruções RISC fazem menos por instrução do que as antigas instruções CISC, o que significa que o problema é ainda mais grave do que parece. A conclusão é a seguinte: atualmente, há menos tempo disponível para o processamento de protocolos do que havia antes, e assim os protocolos têm de se tornar mais simples.

Um terceiro problema é que o protocolo go back n tem um desempenho ruim em linhas com um grande produto da largura de banda pelo retardo. Por exemplo, considere uma linha de 4000 km operando a 1 Gbps. O tempo de transmissão de ida e volta é de 40 ms, e nesse tempo um transmissor pode enviar 5 megabytes. Se for detectado algum erro, serão necessários pelo menos 40 ms para que o transmissor seja informado. Se for utilizado o go back n, o transmissor terá de enviar novamente não só o pacote com problemas, mas também os 5 megabytes de pacotes subseqüentes. É claro que isso é um maciço desperdício de recursos.

Um quarto problema é que as linhas de gigabits são fundamentalmente diferentes das linhas de megabits, pois as longas linhas de gigabits são limitadas pelo retardo e não pela largura da banda. Na Figura 6.47, mostramos o tempo necessário para transferir um arquivo de 1 megabit por 4000 km em diferentes velocidades de transmissão. Com velocidades de até 1 Mbps, o tempo é dominado pela taxa com que os bits podem ser enviados. A 1 Gbps, o retardo de 40 ms da viagem de ida e volta domina o tempo de 1 ms necessário para colocar os bits no cabo de fibra óptica. Aumentos adicionais na largura de banda praticamente não têm nenhum efeito.

A Figura 6.47 traz implicações desagradáveis para protocolos de redes. Ela mostra que os protocolos stop-and-wait, como RPC, têm um limite superior

inerente em seu desempenho. Esse limite é ditado pela velocidade da luz. Não há progresso tecnológico em termos de óptica que possa melhorar isso (entretanto, novas leis da física poderiam ajudar).

Um quinto problema que vale a pena mencionar não é tecnológico ou proveniente de um protocolo como os outros, e sim um resultado de novas aplicações. Em termos simples, trata-se do problema de muitas aplicações de gigabits, como as de multimídia, em que a variância nos tempos de chegada de pacotes é tão importante quanto o próprio retardo médio. Uma taxa de transferência baixa, porém constante, quase sempre é preferível a uma taxa rápida, porém irregular.

Passaremos agora dos problemas aos meios para lidar com eles. Primeiro, faremos alguns comentários gerais, depois examinaremos mecanismos de protocolos, layout de pacotes e software de protocolo.

O princípio básico que todos os projetistas de redes de gigabits deveriam saber de cor é:

Crie projetos para obter mais velocidade e não para otimizar a largura de banda.

[arte: ver original p. 571]

[Dísticos]

[1] Tempo de transferência de arquivos

[2] 1000 s

100 s

10 s

1 s

100 ms

10 ms

1 ms

[3] 10^3 10^4 10^5 10^6 10^7 10^8 10^9 10^{10} 10^{11} 10^{12}

[4] Taxa de dados (bps)

[F]Figura 6.47

[FL] Tempo de transferência e confirmação de um arquivo de 1 megabit por uma linha de 4000 km

Com freqüência, os protocolos antigos eram projetados para minimizar o número de bits no cabo, quase sempre usando pequenos campos e agrupando-os em bytes e palavras. Hoje em dia, há largura de banda suficiente. O problema é o processamento dos protocolos, e assim os protocolos devem ser projetados com o objetivo de minimizá-lo. Os projetistas do IPv6 sem dúvida entenderam esse princípio.

Uma maneira muito interessante de aumentar a velocidade é criar interfaces de rede rápidas em hardware. A dificuldade é que, a menos que o protocolo seja muito simples, o hardware significa apenas uma placa complementar com uma segunda CPU e seu próprio programa. Para evitar que o co-processador de rede seja mais caro que a CPU principal, muitas vezes ele é um chip mais lento. A conseqüência desse projeto é que, durante grande parte do tempo, a CPU principal (rápida) está ociosa, esperando que a segunda CPU (lenta) execute o trabalho crítico. É um mito pensar que a CPU principal tem outro trabalho a fazer enquanto espera. Além disso, quando duas CPUs de uso geral se comunicam podem ocorrer condições de competição, e portanto são necessários protocolos elaborados entre os dois processadores, a fim de sincronizá-los corretamente. Em geral, a melhor estratégia é simplificar os protocolos e fazer com que a CPU principal realize o trabalho.

Agora, vamos examinar a questão do feedback em protocolos de alta velocidade. Devido ao (relativamente) longo retardo no loop, o feedback deve ser evitado, pois o tempo necessário para o receptor enviar um sinal ao transmissor é muito longo. Um exemplo de feedback é o controle da taxa de transmissão pela

utilização de um protocolo de janela deslizante. Para evitar os (longos) retardos inerentes ao fato do receptor enviar atualizações de janelas ao transmissor, é melhor usar um protocolo baseado na taxa. Nesse protocolo, o transmissor pode enviar tudo o que quiser, desde que nunca o faça com maior rapidez que alguma taxa previamente estabelecida por ele e pelo receptor.

Um segundo exemplo de feedback é o algoritmo de início lento de Jacobson. Esse algoritmo promove várias sondagens para verificar quanto a rede pode manipular. Em uma rede de alta velocidade, fazer meia dúzia de pequenas sondagens para ver como a rede se comporta desperdiça um grande volume de largura de banda. Um esquema mais eficiente é fazer com que o transmissor, o receptor e a rede reservem os recursos necessários no momento de estabelecer a conexão. Reservar recursos antecipadamente também oferece a vantagem de tornar mais fácil reduzir a flutuação. Em resumo, buscar altas velocidades, leva o projeto inexoravelmente em direção a uma operação orientada a conexões, ou algo bem parecido. É claro que, se a largura de banda se tornar tão abundante no futuro que ninguém se preocupe com o fato de desperdiçar grandes quantidade desse recurso, as regras de projeto se tornarão muito diferentes.

O layout de pacotes é uma consideração importante nas redes de gigabits. O cabeçalho deve conter o mínimo possível de campos, a fim de reduzir o tempo de processamento. Esses campos devem ser grandes o suficiente para realizar o trabalho e ter alinhamento de palavras com a finalidade de facilitar o processamento. Nesse contexto, "grande o suficiente" significa que não ocorrerão problemas como repetição de números de seqüência enquanto ainda existem pacotes antigos, receptores incapazes de anunciar espaço de janela suficiente porque o campo da janela é muito pequeno e assim por diante.

O cabeçalho e os dados devem conter totais de verificação distintos, por dois motivos. Primeiro, para que seja possível executar um total de verificação do

cabeçalho e não dos dados. Segundo, para verificar se o cabeçalho está correto antes de começar a copiar os dados para o espaço do usuário. O ideal é executar o total de verificação com os dados no momento em que eles forem copiados para o espaço do usuário; no entanto, se o cabeçalho estiver incorreto, a cópia poderá ser feita para o processo errado. Para evitar uma cópia incorreta, mas permitir que o total de verificação dos dados seja calculado durante a cópia, é essencial que haja dois totais de verificação distintos.

O tamanho máximo dos dados deve ser grande, de modo a permitir uma operação eficiente mesmo na presença de retardos longos. Além disso, quanto maior for o bloco de dados, menor será a fração da largura de banda total destinada aos cabeçalhos. 1500 bytes é um tamanho muito pequeno.

Outro recurso valioso é a possibilidade de enviar um volume normal de dados junto com a solicitação de conexão. Desse modo, é possível economizar o tempo de uma viagem de ida e volta.

Por fim, é interessante fazer algumas considerações sobre o software do protocolo. Um procedimento fundamental é concentrar-se em um caso bem-sucedido. Muitos protocolos antigos tendem a enfatizar o que fazer se algo der errado (por exemplo, quando um pacote é perdido). Para tornar os protocolos mais rápidos, o projetista deve se empenhar em minimizar o tempo de processamento quando tudo estiver correto. Minimizar o tempo de processamento quando ocorrer algum erro tem importância secundária.

Uma segunda questão relacionada ao software é minimizar o tempo de cópia. Como vimos antes, a cópia de dados é quase sempre a maior fonte de overhead. O ideal é que o hardware deposite todos os pacotes recebidos na memória como um bloco contíguo de dados. O software deve então copiar esse pacote para o buffer do usuário, em um único bloco. Dependendo do funcionamento do cache, talvez até seja interessante evitar um loop de cópia. Ou seja, para copiar 1024

palavras, o modo mais rápido pode ser usar 1024 instruções MOVE seguidas (ou 1024 pares carga-armazenamento). A rotina de cópia é tão importante que deveria ser cuidadosamente desenvolvida em código assembly, a menos que haja uma maneira de fazer o compilador produzir exatamente o código ótimo.

[T2] 6.7 Resumo

A camada de transporte é a chave para a compreensão dos protocolos em camadas. Ela oferece vários serviços, dos quais o mais importante é um fluxo de bytes confiável, orientado à conexões e fim a fim, do transmissor até o receptor. Ela é acessada por meio de primitivas de serviço que permitem o estabelecimento, o uso e o encerramento de conexões. Uma interface comum da camada de transporte é oferecida pelos soquetes de Berkeley.

Os protocolos de transporte devem ser capazes de realizar o gerenciamento de conexões em redes não confiáveis. O estabelecimento de conexões é complicado pela existência de duplicatas de pacotes atrasadas que podem reaparecer em momentos inoportunos. Para lidar com elas, são necessários handshakes de três vias para estabelecer conexões. O encerramento de uma conexão é um pouco mais fácil, mas ainda assim está longe de ser uma questão trivial, devido ao problema dos dois exércitos.

Mesmo quando a camada de rede é totalmente confiável, a camada de transporte tem muito trabalho a fazer. Ela deve manipular todas as primitivas de serviço, gerenciar as conexões e os timers, e ainda alocar e utilizar créditos.

A Internet tem dois protocolos de transporte importantes: UDP e TCP. O UDP é o protocolo sem conexões usado principalmente com envoltória para pacotes IP com o recurso adicional de multiplexar e demultiplexar vários processos utilizando um único endereço IP. O UDP pode ser usado, por exemplo, em interações cliente/servidor, empregando RPC. Ele também pode ser usado na

construção de protocolos de tempo real, como o RTP.

O principal protocolo de transporte da Internet é o TCP, que fornece um fluxo de bytes bidirecional confiável. Ele utiliza um cabeçalho de 20 bytes em todos os segmentos. Os segmentos podem ser divididos pelos roteadores da Internet; portanto, os hosts devem estar preparados para reorganizá-los. Um grande trabalho tem sido realizado na tentativa de otimizar o desempenho do TCP, com os algoritmos de Nagle, Clark, Jacobson, Karn e outros. As redes sem fios acrescentam uma variedade de complicações ao TCP. O TCP transacional é uma extensão do TCP que manipula interações cliente/servidor com um número reduzido de pacotes.

Em geral, o desempenho da rede é dominado pelo protocolo e pelo overhead de processamento de TPDUs, e essa situação piora em velocidades mais elevadas. Os protocolos devem ser projetados para minimizar o número de TPDUs, as mudanças de contexto e o número de vezes que cada TPDU é copiada. Para redes de gigabits, são mais aconselháveis protocolos simples.

[T2] Problemas

1. Em nosso exemplo de primitivas de transporte da Figura 6.2, a primitiva LISTEN é uma chamada de bloqueio. Ela é estritamente necessária? Caso não seja, explique como uma primitiva sem bloqueio poderia ser usada. Que vantagem isso ofereceria em relação ao esquema descrito no texto?

2. No modelo básico da Figura 6.4, parte-se do pressuposto de que os pacotes podem se perder na camada de rede e, por isso, devem ser confirmados individualmente. Suponha que a camada de rede seja 100% confiável e nunca perca pacotes. Que mudanças, se houver, são necessárias na Figura 6.4?

3. Em ambas as partes da Figura 6.6, existe um comentário de que o valor de *SERVER_PORT* deve ser igual no cliente e no servidor. Por que isso é tão

importante?

4. Suponha que o esquema baseado no clock empregado na geração de números de seqüência iniciais seja usado com um contador de clock de 15 bits de largura. O clock pulsa a cada 100 ms, e o tempo de vida máximo de cada pacote é de 60 s. Com que freqüência a ressincronização deve ser feita:

(a) Na pior das hipóteses?

(b) Quando os dados consumirem 240 números de seqüência/minuto?

5. Por que o tempo máximo de duração de um pacote *T* deve ser longo o suficiente para assegurar que não apenas o pacote mas também todas as suas confirmações tenham desaparecido?

6. Imagine que, em vez do handshake de três vias, tenha sido utilizado um handshake de duas vias para estabelecer uma conexão. Em outras palavras, a terceira mensagem não foi solicitada. É possível que agora ocorra um impasse? Forneça um exemplo ou mostre que não existe nenhum.

7. Imagine um problema genérico dos *n* exércitos, no qual um acordo entre dois quaisquer dos exércitos azuis é suficiente para a vitória. Existe algum protocolo que permita ao exército azul vencer?

8. Considere o problema da recuperação do funcionamento depois que panes nos hosts (Figura 6.18). Se o intervalo entre a gravação e o envio de uma confirmação, ou vice-versa, pode se tornar relativamente pequeno, quais são as duas melhores estratégias transmissor/receptor para minimizar a possibilidade de uma falha do protocolo?

9. É possível a ocorrência de impasses com a entidade de transporte descrita no texto (Figura 6.20)?

10. Por curiosidade, o responsável pela implementação da entidade de transporte da Figura 6.20 decidiu colocar contadores no interior do procedimento *sleep* para coletar informações estatísticas sobre o array *conn*. Dentre elas estão o número

de conexões de cada um dos sete estados possíveis, n_i (i = 1, ..., 7). Depois de elaborar um longo programa em FORTRAN para analisar os dados, nosso implementador descobriu que a relação [ver símbolo]n_i = *MAX_CONN* parece ser sempre verdadeira. Existem outras características invariantes envolvendo somente essas sete variáveis?

11. O que acontece quando o usuário da entidade de transporte ilustrada na Figura 6.20 envia uma mensagem de tamanho zero? Discuta o significado da sua resposta.

12. Para cada evento possível na entidade de transporte da Figura 6.20, diga se ele é válido quando o usuário está inativo no estado *sending*.

13. Discuta as vantagens e desvantagens dos protocolos de créditos em comparação com os protocolos de janela deslizante.

14. Por que o UDP existe? Não teria sido suficiente deixar que os processos dos usuários enviassem pacotes IP brutos?

15. Considere um protocolo simples no nível da aplicação, elaborado sobre o UDP e que permite a um cliente recuperar um arquivo de um servidor remoto que reside em um endereço conhecido. Primeiro, o cliente envia uma solicitação com o nome do arquivo, e o servidor responde com uma seqüência de pacotes de dados contendo partes diferentes do arquivo solicitado. Para assegurar confiabilidade e entrega em seqüência, o cliente e o servidor utilizam um protocolo stop-and-wait. Ignorando a questão óbvia do desempenho, você percebe algum problema com esse protocolo? Pense cuidadosamente na possibilidade de panes em processos.

16. Um cliente transmite uma solicitação de 128 bytes a um servidor localizado a 100 km de distância sobre um cabo de fibra óptica de 1 gigabit. Qual é a eficiência da linha durante a chamada de procedimento remoto?

17. Considere a situação do problema anterior novamente. Calcule o menor

tempo de resposta possível para a linha de 1 Gbps e para uma linha de 1 Mbps. Que conclusão é possível tirar desses dados?

18. Tanto o UDP quanto o TCP empregam números de portas para identificar a entidade de destino ao entregarem uma mensagem. Forneça duas razões pelas quais esses protocolos criaram uma nova ID abstrata (números de portas), em vez de usarem IDs de processos, que já existiam quando esses protocolos foram projetados.

19. Qual é o tamanho total da MTU mínima do TCP, incluindo o overhead do TCP e do IP, mas sem incluir o overhead da camada de enlace de dados?

20. A fragmentação e a remontagem de datagramas são tratadas pelo IP e são invisíveis para o TCP. Isso quer dizer que o TCP não tem de se preocupar com a chegada de dados na ordem errada?

21. O RTP é usado para transmitir áudio com qualidade de CD, que gera um par de amostras de 16 bits 44.100 vezes/s, uma amostra para cada um dos canais estereofônicos. Quantos pacotes por segundo o RTP deve transmitir?

22. Seria possível inserir o código do RTP no núcleo do sistema operacional, juntamente com o código do UDP? Explique sua resposta.

23. A um processo no host 1 foi atribuído à porta *p*, e um processo no host 2 foi atribuído à porta *q*. É possível haver duas ou mais conexões TCP entre essas duas portas ao mesmo tempo?

24. Na Figura 6.29 vimos que, além do campo *Acknowledgement* de 32 bits, existe um bit *ACK* na quarta palavra. Isso realmente acrescenta algo? Por que, ou por que não?

25. A carga útil máxima de um segmento TCP é 65.515 bytes. Por que foi escolhido um número tão estranho?

26. Descreva duas maneiras de entrar no estado *SYN RCVD* da Figura 6.33.

27. Apresente uma desvantagem potencial da utilização do algoritmo de Nagle

em uma rede muito congestionada.

28. Considere o efeito de usar o início lento em uma linha com um tempo de percurso de ida e volta de 10 ms e sem congestionamento. A janela de recepção tem 24 KB e o tamanho máximo do segmento é 2 KB. Quanto tempo é necessário para que a primeira janela completa possa ser enviada?

29. Suponha que a janela de congestionamento do TCP seja definida como 18 KB e que ocorra um timeout. Qual será o tamanho da janela se as próximas quatro rajadas de transmissão forem bem-sucedidas? Suponha que o tamanho máximo do segmento seja 1 KB.

30. Se o tempo de viagem de ida e volta no TCP, denominado *RTT*, for igual a 30 ms, e se as confirmações seguintes chegarem depois de 26, 32 e 24 ms, respectivamente, qual será a nova estimativa para *RTT* empregando-se o algoritmo de Jacobson? Considere $\alpha = 0{,}9$.

31. Uma máquina TCP está transmitindo janelas completas de 65.535 bytes através de um canal de 1 Gbps que tem um retardo de 10 ms em um dos sentidos. Qual é o throughput máximo que é possível alcançar? Qual é a eficiência da linha?

32. Qual é a maior velocidade da linha em que um host pode transmitir cargas úteis do TCP de 1500 bytes com uma duração máxima de pacote de 120 segundos, sem que os números de seqüência se repitam? Leve em conta o overhead do TCP, do IP e da Ethernet. Suponha que os quadros Ethernet possam ser enviados continuamente.

33. Em uma rede com um tamanho máximo de TPDU igual a 128 bytes, tempo máximo de duração de uma TPDU igual a 30 s e um número de seqüência de 8 bits, qual é a taxa máxima de transferência de dados por conexão?

34. Suponha que você esteja medindo o tempo necessário para receber uma TPDU. Quando ocorrer uma interrupção, você lerá o clock do sistema em

milissegundos. Quando a TPDU tiver sido completamente processada, você lerá o clock mais uma vez. O valor 0 ms é medido 270.000 vezes, e 1 ms é medido 730.000 vezes. Quanto tempo é necessário para receber uma TPDU?

35. Uma CPU executa instruções a uma velocidade de 1000 MIPS. Os dados podem ser copiados 64 bits de cada vez, sendo necessárias dez instruções para copiar cada palavra. Se um pacote recebido tiver de ser copiado duas vezes, esse sistema será capaz de tratar uma linha de 1 Gbps? Para simplificar, suponha que todas as instruções, mesmo aquelas que lêem ou gravam na memória, funcionam a uma velocidade máxima de 1000 MIPS.

36. Para contornar o problema da repetição dos números de seqüência enquanto os pacotes antigos ainda existem, é possível utilizar números de seqüência de 64 bits. Contudo, um cabo de fibra óptica pode utilizar uma velocidade de 75 Tbps, pelo menos teoricamente. Qual é o tempo máximo de duração dos pacotes necessário para garantir que as redes de 75 Tbps do futuro não terão problemas de repetição dos números de seqüência, mesmo com números de seqüência de 64 bits? Suponha que cada byte tenha seu próprio número de seqüência, como acontece no TCP.

37. Forneça uma vantagem da RPC no UDP sobre o TCP transacional. Apresente uma vantagem do T/TCP sobre a RPC.

38. Na Figura 6.40(a), vemos que são necessário 9 pacotes para completar a RPC. Existe alguma circunstância em que sejam necessários exatamente 10 pacotes?

39. Na Seção 6.6.5, calculamos que uma linha de gigabits descarrega 80.000 pacotes/s no host, dando a ele apenas 6250 instruções para processar e deixar metade do tempo da CPU para aplicações. Esse cálculo partiu do princípio de que cada pacote tem 4 KB. Refaça os cálculos para um pacote com o tamanho dos pacotes ARPANET (128 bytes). Em ambos os casos, suponha que os tamanhos dos pacotes dados incluem todo o overhead.

40. Para uma rede de 1 Gbps operando por 4000 km, o retardo é o fator limitante, e não a largura da banda. Considere uma MAN com uma distância média entre a origem e o destino de 20 km. Em qual taxa de dados o retardo do percurso de ida e volta devido a velocidade da luz é igual ao retardo da transmissão para um pacote de 1 KB?

41. Calcule o produto da largura de banda pelo retardo para as redes a seguir: (1) T1 (1,5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps) e (4) STS-3 (155 Mbps). Suponha um RTT de 100 ms. Lembre-se de que um cabeçalho do TCP tem 16 bits reservados para Window Size. Quais são as implicações, de acordo com seus cálculos?

42. Qual é o produto do retardo pela largura de banda para um canal de 50 Mbps em um satélite geoestacionário? Se todos os pacotes forem de 1500 bytes (incluindo o overhead), qual deverá o tamanho da janela, medido em pacotes?

43. O servidor de arquivos da Figura 6.6 está longe de ser perfeito e poderia usar alguns aperfeiçoamentos. Faça as seguintes modificações:

(a) Dê ao cliente um terceiro argumento que especifique um intervalo de bytes.

(b) Adicione um flag do cliente -w que permita a gravação do arquivo no servidor.

44. Modifique o programa da Figura 6.20 para executar uma recuperação de erros. Acrescente um novo tipo de pacote, *reset*, que pode chegar depois que uma conexão é aberta por ambos os lados, sem ser fechada por nenhum deles. Esse evento, que ocorre simultaneamente nas duas extremidades da conexão, significa que os pacotes que estavam em trânsito foram entregues ou destruídos, mas nos dois casos não estão mais na sub-rede.

45. Crie um programa que simule o gerenciamento de buffers em uma entidade de transporte usando uma janela deslizante para o controle do fluxo, em lugar do sistema de créditos da Figura 6.20. Deixe que os processos das camadas mais altas abram conexões ao acaso, que transmitam dados e encerrem as conexões.

Para simplificar, faça com que todos os dados passem da máquina *A* para a máquina *B*, e nunca o contrário. Experimente diferentes estratégias de alocação de buffer na máquina *B*, como dedicar buffers a conexões específicas em comparação com um conjunto comum de buffers, e meça o throughput total alcançado em cada estratégia.

46. Projete e implemente um sistema de bate-papo que permita a comunicação entre vários grupos de usuários. Um coordenador de bate-papo reside em um endereço de rede conhecido, utiliza o UDP para comunicação com clientes de bate-papo, instala servidores de bate-papo para cada sessão de bate-papo e mantém um diretório de sessão de bate-papo. Existe um servidor de bate-papo por sessão. Um servidor de bate-papo usa o TCP para se comunicar com clientes. Um cliente de bate-papo permite que usuários iniciem, se reúnam e saiam de uma sessão de bate-papo. Projete e implemente o código do coordenador, do servidor e do cliente.

[TA1]Capítulo

[TA2]7

[T1]A camada de aplicação

Depois de passarmos por todas as camadas preliminares, chegamos à camada em que são encontradas todas as aplicações. As camadas situadas abaixo da camada de aplicação têm a função de oferecer um serviço de transporte confiável mas, na verdade, elas não executam qualquer tarefa para os usuários. Neste capítulo, estudaremos algumas aplicações reais de redes.

No entanto, mesmo na camada de aplicação existe a necessidade de protocolos de suporte, a fim de permitir que as aplicações funcionem. Da mesma forma, examinaremos um desses protocolos antes de iniciarmos o estudo das aplicações em si. O item em questão é o DNS, que cuida da nomenclatura na Internet. Depois disso, examinaremos três aplicações reais: correio eletrônico, a World Wide Web e, por fim, multimídia.

[T2] 7.1 DNS — Domain Name System

Embora os programas teoricamente possam se referir a hosts, caixas de correio e outros recursos utilizando seus endereços binários de rede (por exemplo, endereços IP), esses endereços são difíceis de memorizar. Além disso, enviar correio eletrônico para *tana*@*128.111.24.41* significa que, se o ISP ou a organização de Tana muar o servidor de correio para uma máquina diferente com outro endereço IP, seu endereço de correio eletrônico terá de mudar. Conseqüentemente, foram introduzidos nomes em ASCII para desacoplar os nomes das máquinas dos endereços dessas máquinas. Desse modo, o endereço de Tana poderia ser algo como *tana*@*art.uscb.edu*. Todavia, a rede em si só reconhecer endereços numéricos; portanto, é necessário algum tipo de

mecanismo para converter os strings ASCII em endereços de rede. Nas seções a seguir, estudaremos como esse mapeamento é feito na Internet.

Na ARPANET, havia simplesmente um arquivo, *hosts.txt*, que listava todos os hosts e seus endereços IP. Toda noite, esse arquivo era acessado por todos os hosts no local em que era mantido. Para uma rede de algumas centenas de grandes máquinas de tempo compartilhado, essa estratégia funcionava razoavelmente bem.

No entanto, quando milhares de minicomputadores e PCs foram conectadas à rede, todos perceberam que essa estratégia não poderia continuar a ser utilizada para sempre. Por um lado, o arquivo acabaria por se tornar grande demais. No entanto, a razão mais importante é que poderia haver conflitos de nomes de hosts constantemente, a menos que os nomes fossem gerenciados de uma forma centralizada, algo totalmente fora de cogitação em uma enorme rede internacional, devido à carga e à latência. Para resolver esses problemas, foi criado o **DNS (Domain Name System — sistema de nomes de domínios).**

A essência do DNS é a criação de um esquema hierárquico de atribuição de nomes baseado no domínio e de um sistema de bancos de dados distribuídos para implementar esse esquema de nomenclatura. Ele é usado principalmente para mapear nomes de hosts e destinos de mensagens de correio eletrônico em endereços IP, mas também pode ser usado para outros objetivos. O DNS é definido nas RFCs 1034 e 1035.

Em resumo, o DNS é utilizado da forma descrita a seguir. Para mapear um nome em um endereço IP, um programa aplicativo chama um procedimento de biblioteca denominado **resolvedor** e repassa a ele o nome como um parâmetro. Vimos um exemplo de resolvedor, *gethostbyname*, na Figura 6.6. O resolvedor envia um pacote UDP a um servidor DNS local, que procura o nome e retorna o endereço IP ao resolvedor. Em seguida, o resolvedor retorna o endereço IP ao

programa aplicativo que fez a chamada. Munido do endereço IP, o programa pode então estabelecer uma conexão TCP com o destino ou enviar pacotes UDP até ele.

[T3] 7.1.1 O espaço de nomes do DNS

Gerenciar um grande conjunto de nomes que está constantemente mudando não é um problema de fácil resolução. Em um sistema postal, o gerenciamento de nomes é feito através do uso de letras que especificam (implícita ou explicitamente), o país, o estado ou a província, a cidade e a rua do destinatário. Através do uso desse tipo de endereçamento hierárquico, não há confusão entre o João da Silva que mora na Rua Barata Ribeiro, São Paulo, e o João da Silva que mora na Rua Barata Ribeiro, Rio de Janeiro. O DNS funciona da mesma forma. Conceitualmente, a Internet é dividida em mais de 200 **domínios** de nível superior, onde cada domínio cobre muitos hosts. Cada domínio é particionado em subdomínios, que também são particionados e assim por diante. Todos esses domínios podem ser representados por uma árvore, como mostra a Figura 7.1. As folhas da árvore representam domínios que não têm subdomínios (mas que contêm máquinas, é claro). Um domínio folha contém um único host ou pode representar uma empresa e conter milhares de hosts.

Existem dois tipos de domínios de nível superior: genéricos e de países. Os domínios genéricos originais eram *com* (comercial), *edu* (instituições educacionais), *gov* (instituições governamentais), *int* (certas organizações internacionais), *mil* (órgãos das forças armadas), *net* (provedores de rede) e *org* (organizações sem fins lucrativos). Os domínios de países incluem uma entrada para cada país, conforme a definição da ISO 3166.

[arte: ver original p. 581]

[Dísticos]

[1] Genéricos

[2] Países

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 7.1

[FL] Uma parte do espaço de nomes de domínios da Internet

Em novembro de 2000, a ICANN aprovou quatro novos domínios de nível superior e de uso geral, isto é, *biz* (negócios), *info* (informações), *name* (nomes de pessoas) e *pro* (profissões, como médicos e advogados). Além disso, foram introduzidos mais três domínios de nível superior especializados a pedido de certas indústrias. Esses domínios são *aero* (indústria aeroespacial), *coop* (cooperativas) e *museum* (museus). Outros domínios de nível superior serão acrescentados no futuro.

A propósito, à medida que a Internet se torna mais comercial, ela também se torna mais competitiva. Por exemplo, veja o domínio *pro*. Ele se destina a profissionais certificados. Porém, o que é um profissional? Além disso, um profissional é certificado por quem? Médicos e advogados sem dúvida são profissionais. Contudo, e os fotógrafos autônomos, os professores de piano, os mágicos, os encanadores, os barbeiros, os exterminadores, os tatuadores, os artistas, os mercenários e as prostitutas? Todas essas são ocupações profissionais e, portanto, podem obter domínios *pro*? Nesse caso, quem irá certificar os profissionais liberais individuais?

Em geral, é fácil obter um domínio de segundo nível, como *nome-da-empresa*.com. Isso exige apenas um registro do domínio de nível superior correspondente (nesse caso, *com*) para verificar se o nome desejado está disponível e não é marca registrada de outra pessoa. Se não houver nenhum

problema, o solicitante pagará uma pequena taxa anual e conseguirá o nome. Até agora, virtualmente toda palavra comum (em inglês) é usada no domínio *com*. Experimente pesquisar utensílios domésticos, animais, plantas, partes do corpo etc. Quase todas as palavras são usadas.

Cada domínio tem seu nome definido pelo caminho ascendente entre ele e a raiz (sem nome). Esses componentes são separados por pontos. Dessa forma, o departamento de engenharia da Sun Microsystems poderia ser *eng.sun.com.*, em vez de um nome no estilo UNIX, como */com/sun/eng.* Observe que essa nomenclatura hierárquica significa que *eng.sun.com.* não entra em conflito com um possível uso de *eng* em *eng.yale.edu.*, que poderia ser usado pelo departamento de língua inglesa de Yale University.

Os nomes de domínios podem ser absolutos ou relativos. Um nome de domínio absoluto termina com um ponto (por exemplo, *eng.sun.com.*), ao contrário de um nome de domínio relativo. Os nomes relativos têm de ser interpretados em algum contexto para determinar exclusivamente seu verdadeiro significado. Em ambos os casos, um nome de domínio se refere a um nó especifico da árvore e a todos os nós abaixo dele.

Os nomes de domínios não fazem distinção entre letras maiúsculas e minúsculas. Portanto, *edu*, *Edu* e *EDU* têm o mesmo significado. Os nomes de componentes podem ter até 63 caracteres, e os nomes de caminhos completos não podem exceder 255 caracteres.

Em princípio, os domínios podem ser inseridos na árvore de duas formas distintas. Por exemplo, *cs.yale.edu* poderia ser igualmente listado sob o domínio de país *us* como *cs.yale.ct.us*. Contudo, na prática, quase todas as organizações dos Estados Unidos estão sob um domínio genérico e, praticamente, todas fora dos Estados Unidos estão sob o domínio de seu país. Não existe regra contra o registro sob dois domínios de nível superior, mas poucas organizações além das

multinacionais o fazem (por exemplo, *sony.com* e *sony.nl*).

Cada domínio controla como serão alocados todos os domínios que estão abaixo dele. Por exemplo, o Japão tem os domínios *ac.jp* e *co.jp* que espelham *edu* e *com*. A Holanda não faz essa distinção e coloca todas as organizações diretamente sob *nl*. Os três domínios a seguir representam departamentos de ciência da computação de universidades:

1. *cs.yale.edu* (Yale University, Estados Unidos).
2. *cs.vu.nl* (Vrije Universiteit, Holanda).
3. *cs.keio.ac.jp* (Keio University, Japão).

Para que um novo domínio seja criado, é necessária a permissão do domínio no qual ele será incluído. Por exemplo, se o grupo VLSI tiver começado em Yale e quiser ser conhecido como *vlsi.cs.yale.edu*, ele precisará da permissão de quem gerencia *cs.yale.edu*. Da mesma forma, se uma nova universidade for licenciada, digamos a University of Northern South Dakota, ela terá de solicitar ao gerente do domínio *edu* que lhe atribua o domínio *unsd.edu*. Dessa forma, os conflitos de nomes são evitados e cada domínio pode controlar seus subdomínios. Uma vez que um novo domínio tenha sido criado e registrado, ele poderá criar subdomínios, tais como *cs.unsd.edu*, sem que seja necessária a permissão de alguém que esteja em um nível mais alto na árvore.

A atribuição de nomes leva em consideração as fronteiras organizacionais, e não as redes físicas. Por exemplo, mesmo que os departamentos de ciência da computação e de engenharia elétrica estejam localizados no mesmo prédio e compartilhem a mesma LAN, eles poderão ter domínios distintos. Da mesma forma, mesmo que o departamento de ciência da computação esteja dividido em dois prédios, normalmente todos os hosts instalados em ambos pertencerão ao mesmo domínio.

[T3] 7.1.2 Registros de recursos

Todo domínio, independente de ser um único host ou um domínio de nível superior, pode ter um conjunto de **registros de recursos** associado a ele. Para um único host, o registro de recurso mais comum é apenas seu endereço IP, mas também existem muitos outros tipos de registros de recursos. Quando um resolvedor repassa um nome de domínio ao DNS, o que ele obtém são os registros de recursos associados ao nome em questão. Portanto, a principal função do DNS é mapear nomes de domínios em registros de recursos.

Um registro de recurso é uma tupla de cinco campos. Apesar de serem codificados em binário para proporcionar maior eficiência, na maioria das exposições, os registros de recursos são mostrados como texto ASCII, uma linha para cada registro de recurso. O formato que utilizaremos é:

Domain_name Time_to_live Class Type Value

Domain_name informa o domínio ao qual esse registro se aplica. Normalmente, existem muitos registros para cada domínio, e cada cópia do banco de dados armazena informações sobre vários domínios. Assim, esse campo é a chave de pesquisa primária utilizada para atender às consultas. A ordem dos registros no banco de dados não é significativa.

Time_to_live fornece uma indicação da estabilidade do registro. As informações muito estáveis são definidas com um número alto, como 86.400 (o número de segundos em 1 dia). As informações muito voláteis recebem um número baixo, como 60 (1 minuto). Voltaremos a esse ponto mais adiante, quando discutirmos os caches.

O terceiro campo de cada registro de recurso é *Class*. No caso de informações relacionadas à Internet, ele é sempre *IN*. Para informações não relacionadas à Internet, podem ser empregados outros códigos; porém, esses outros códigos raramente são encontrados na prática.

O campo *Type* informa qual é o tipo do registro. Os tipos mais importantes estão listados na Figura 7.2.

[arte: ver original p. 583]

[T]Tabela

Tipo

SOA

A

MX

NS

CNAME

PTR

HINFO

TXT

Significado

Início de autoridade

Endereço IP de um host

Troca de mensagens de correio

Servidor de nomes

Nome canônico

Ponteiro

Descrição de host

Texto

Valor

Parâmetros para essa zona

Inteiro de 32 bits

Prioridade, domínio disposto a aceitar correio eletrônico

Nome de um servidor para este domínio

Nome de domínio

Nome alternativo de um endereço IP

CPU e sistema operacional em ASCII

Texto ASCII não interpretado

[F]Figura 7.2

[FL] Os principais tipos de registros de recursos do DNS para o IPv4

Um registro *SOA* fornece o nome da principal fonte de informações sobre a zona do servidor de nomes (descrita a seguir), o endereço de correio eletrônico do administrador, um número de série exclusivo e diversos flags e timeouts.

O tipo de registro mais importante é o registro *A* (Address). Ele contém um endereço IP de 32 bits de algum host. Todos os hosts da Internet devem ter pelo menos um endereço IP, de forma que outras máquinas possam se comunicar com ele. Alguns hosts têm duas ou mais conexões de rede; nesse caso, eles terão um registro de recurso do tipo *A* por conexão de rede (e, portanto, por endereço IP). O DNS pode ser configurado para circular por esses endereços, retornando o primeiro registro na primeira solicitação, o segundo registro na segunda solicitação e assim por diante.

O próximo tipo de registro mais importante é o registro *MX*. Ele especifica o nome do host preparado para aceitar mensagens de correio eletrônico para o domínio especificado. O registro *MX* é utilizado porque nem toda máquina está preparada para aceitar correio eletrônico. Se alguém quiser enviar correio eletrônico para *bill*@*microsoft.com*, o host transmissor precisará encontrar um servidor de correio em *microsoft.com* que esteja disposto a aceitar correio eletrônico. O registro *MX* pode fornecer essa informação.

Os registros *NS* especificam servidores de nomes. Por exemplo, todos os bancos de dados DNS têm um registro *NS* para cada um dos domínios de nível superior;

assim, as mensagens de correio eletrônico podem ser enviadas para partes distantes da árvore de atribuição de nomes. Voltaremos a esse ponto mais adiante.

Os registros *CNAME* permitem a criação de nomes alternativos. Por exemplo, uma pessoa familiarizada com a atribuição de nomes na Internet em geral que deseja enviar uma mensagem para alguém cujo nome de login seja paul no departamento de ciência da computação do MIT poderá imaginar que *paul@cs.mit.edu* seja o endereço correto. Na realidade, esse endereço não servirá, pois o domínio do departamento de ciência da computação do MIT é *lcs.mit.edu*. No entanto, o MIT poderia criar uma entrada *CNAME* para orientar pessoas e programas na direção correta, oferecendo um serviço para quem não sabe disso. Uma entrada como esta poderia executar essa função:

[TD] cs.mit.edu 86400 IN CNAME lcs.mit.edu [TN]

A exemplo de *CNAME*, *PTR* indica outro nome. No entanto, ao contrário de *CNAME* que, na verdade, é apenas uma definição de macro, *PTR* é um tipo de dados comum do DNS cuja interpretação depende do contexto no qual se encontra. Na prática, essa entrada é quase sempre usada para associar um nome a um endereço IP, a fim de permitir pesquisas de endereços IP e retornar o nome da máquina correspondente. Essas pesquisas são chamadas **pesquisas inversas.**

Os registros *HINFO* permitem que as pessoas descubram a que tipo de máquina e sistema operacional um domínio corresponde. Por fim, os registros *TXT* permitem que os domínios se identifiquem de forma arbitrária. Esses dois tipos de registros são usados para conveniência do usuário. Nenhum deles é obrigatório; portanto, os programas não podem pressupor sua presença (e provavelmente não poderão lidar com eles se os receberem).

Por fim, chegamos ao campo *Value*. Esse campo pode ser um número, um nome de domínio ou um string ASCII. A semântica dependerá do tipo de registro. Na

Figura 7.2, é mostrada uma breve descrição dos campos *Value* de cada um dos principais tipos de registros.

Como exemplo do tipo de informação que se pode encontrar no banco de dados DNS de um domínio, observe a Figura 7.3. Essa figura ilustra parte de um banco de dados (semi-hipotético) para o domínio *cs.vu.nl* mostrado na Figura 7.1. O banco de dados contém sete tipos de registros de recursos.

A primeira linha não destinada a comentários da Figura 7.3 apresenta algumas informações básicas sobre o domínio, que não nos interessarão daqui por diante. As duas linhas seguintes apresentam informações textuais sobre onde o domínio está localizado. Em seguida, vêm duas entradas que mostram a primeira e a segunda opções para a entrega de mensagens de correio eletrônico enviadas para *pessoa@cs.vu.nl*. A entrada *zephyr* (uma máquina específica) deve ser a primeira opção a ser experimentada. Se ela não servir, *top* será a próxima opção.

[arte: ver original p. 585]

[TD]

```
; Authoritative data for cs.vu.nl
cs.vu.nl.         86400   IN   SOA     star boss
(952771,7200,7200,2419200, 86400)
cs.vu.nl.         86400   IN   TXT     "Divisie Wiskunde en
Informatica."
cs.vu.nl.         86400   IN   TXT     "Vrije Universiteit Amsterdam."
cs.vu.nl.         86400   IN   MX      1 zephyr.os.vu.nl.
cs.vu.nl.         86400   IN   MX      2 top.cs.vu.nl.

flits.cs.vu.nl.   86400   IN   HINFO   Sun Unix
flits.cs.vu.nl.   86400   IN   A       130.37.16.112
flits.cs.vu.nl.   86400   IN   A       192.31.231.165
```

flits.cs.vu.nl.	86400	IN	MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	2 zephyr.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	3 top.cs.vu.nl
www.cs.vu.nl.	86400	IN	CNAME	star.os.vu.NL
ftp.cs.vu.nl.	86400	IN	CNAME	zephyr.cs.vu.nl
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
		IN	HINFO	Sun Unix
little-sister		IN	A	130.37.62.23
		IN	HINFO	Mac MacOS
laserjet		IN	A	192.31.231.216
		IN	HINFO	"HP Laserjet IIISi" Proprietary

[TN]

[F]Figura 7.3

[FL] Uma parte de um possível banco de dados DNS para *cs.vu.nl*

Depois da linha em branco, que foi incluída para facilitar a leitura, há linhas informando que *flits* é uma estação de trabalho Sun com o sistema operacional UNIX. Os endereços IP dessa estação também são fornecidos nessas linhas. Em seguida, são oferecidas três opções para tratar as mensagens de correio eletrônico enviadas para *flits.cs.vu.nl*. A primeira delas é naturalmente o próprio *flits*, mas se ele estiver fora do ar, *zephyr* e *top* são a segunda e a terceira opção, respectivamente. Em seguida, há um nome alternativo, *www.cs.vu.nl*, ou seja, um

endereço que pode ser usado sem a necessidade de se especificar uma determinada máquina. A criação desse nome alternativo permite que *cs.vu.nl* modifique seu servidor da World Wide Web sem invalidar o endereço que as pessoas utilizam para acessá-lo. Há um argumento semelhante para *ftp.cs.vu.nl*. As quatro linhas seguintes contêm uma entrada típica para uma estação de trabalho, nesse caso, *rowboat.cs.vu.nl*. As informações fornecidas contêm o endereço IP, as caixas de correio principal e secundária e dados sobre a máquina. Em seguida, vem uma entrada para um sistema não UNIX que não é capaz de receber mensagens de correio eletrônico sozinha, seguida de uma entrada para uma impressora a laser conectada à Internet.

O que não é mostrado (e que não está nesse arquivo) são os endereços IP a serem usados na pesquisa dos domínios de nível superior. Esses endereços são necessários para pesquisar hosts distantes, mas como não fazem parte do domínio *cs.vu.nl*, eles não estão nesse arquivo. Esses endereços são fornecidos pelos servidores raiz, cujos endereços IP estão presentes em um arquivo de configuração do sistema e são carregados para o cache do DNS quando o servidor DNS é inicializado. Existe cerca de uma dezena de servidores raiz espalhados pelo mundo, e que cada um deles conhece os endereços IP de todos os servidores de domínio de nível superior. Desse modo, se uma máquina conhecer o endereço IP de pelo menos um servidor raiz, ela poderá pesquisar qualquer nome DNS.

[T3] 7.1.3 Servidores de nomes

Pelo menos na teoria, um único servidor de nomes poderia conter o banco de dados DNS inteiro e responder a todas as consultas referentes ao banco. Na prática, esse servidor ficaria tão sobrecarregado que seria inútil. Além disso, caso esse servidor viesse a ficar fora do ar, toda a Internet seria atingida.

Para evitar os problemas associados à presença de uma única fonte de

informações, o espaço de nomes do DNS é dividido em **zonas** não superpostas.

Uma forma possível de dividir o espaço de nomes da Figura 7.1 é mostrado na Figura 7.4. Cada zona contém uma parte da árvore e também servidores de nomes que armazenam informações referentes a essa zona. Normalmente, uma zona terá um servidor de nomes principal, que obtém suas informações a partir de um arquivo contido em sua unidade de disco e um ou mais servidores de nomes secundários, que obtêm suas informações a partir do servidor de nomes principal. Para melhorar a confiabilidade, alguns servidores de uma zona podem estar localizados fora dela.

[arte: ver original p. 586]

[Dísticos]

[1] Genérico

[2] Países

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 7.4

[FL] Parte do espaço de nomes do DNS mostrando a divisão em zonas

A localização das fronteiras de uma zona fica a cargo de seu administrador. Essa decisão é tomada principalmente com base no número de servidores de nomes desejados. Por exemplo, na Figura 7.4, Yale tem um servidor para *yale.edu* que trata de *eng.yale.edu*, mas não de *cs.yale.edu*, que é uma zona separada com seus próprios servidores de nomes. Tal decisão pode ser tomada quando um departamento, como língua inglesa, não deseja ter seu próprio servidor de nomes, mas um departamento como ciência da computação deseja tê-lo. Conseqüentemente, *cs.yale.edu* é uma zona separada, mas *eng.yale.edu* não é.

Quando um resolvedor tem uma consulta sobre um nome de domínio, ele a envia a um dos servidores de nomes locais. Se o domínio que estiver sendo procurado estiver sob a jurisdição do servidor de nomes, como *ai.cs.yale.edu* que está sob *cs.yale.edu*, serão retornados os registros de recursos oficiais. Um **registro oficial** é aquele fornecido pela autoridade que gerencia o registro e, portanto, sempre está correto. Os registros mantidos em cache, ao contrário dos registros oficiais, podem estar desatualizados.

No entanto, se o domínio for remoto e não houver informações sobre o domínio solicitado disponíveis no local, o servidor de nomes enviará uma mensagem de consulta para o servidor de nomes de nível superior fazendo perguntas sobre o domínio solicitado. Para tornar esse processo mais claro, considere o exemplo da Figura 7.5. Aqui, um resolvedor localizado em *flits.cs.vu.nl* quer saber o endereço IP do host *linda.cs.yale.edu*. Na etapa 1, ele envia uma consulta ao servidor de nomes local, *cs.vu.nl*. Essa consulta contém o nome de domínio procurado, o tipo (*A*) e a classe (*IN*).

[arte: ver original p. 587]

[Dísticos]

[1]Originador Servidor de nomes VU CS Servidor de nomes Edu Servidor de nomes Yale Servidor de nomes Yale CS

[2]1 2 3 4

[3]flits.cs.vu.nl cs.vu.nl edu-server.net yale.edu cs.yale.edu

[4]8 7 6 5

[F]Figura 7.5

[FL] O modo como um resolvedor procura um nome remoto em oito etapas

Suponha que o servidor de nomes local nunca tenha recebido uma consulta referente a esse domínio e nada saiba sobre ele. O servidor poderá consultar

alguns outros servidores de nomes vizinhos mas, se nenhum deles tiver as informações, ele enviará um pacote UDP ao servidor *edu* fornecido em seu banco de dados (observe a Figura 7.5), *edu-server.net*. Não é provável que esse servidor conheça o endereço de *linda.cs.yale.edu* e *cs.yale.edu*, mas deverá conhecer seus próprios endereços filhos; desse modo, encaminha ao servidor de nomes a solicitação referente a *yale.edu* (etapa 3). Por sua vez, este encaminha a solicitação a *cs.yale.edu* (etapa 4), que deve ter os registros de recursos oficiais. Como cada solicitação é feita por um cliente a um servidor, o registro de recurso solicitado segue o caminho inverso, durante as etapas 5 a 8.

Uma vez que tenham voltado ao servidor de nomes *cs.vu.nl*, esses registros serão incluídos em um cache, caso sejam necessários mais tarde. No entanto, essas informações não são oficiais, pois as alterações em *cs.yale.edu* não serão propagadas para todos os caches do mundo que possam ter conhecimento da situação. Por essa razão, as entradas de cache não terão uma duração muito longa. Por isso, o campo *Time_to_live* é incluído em cada registro de recurso. Ele informa aos servidores de nomes remotos por quanto tempo os registros devem ser mantidos no cache. Se uma determinada máquina já tiver o mesmo endereço IP há anos, talvez seja uma boa idéia manter essas informações no cache durante 1 dia. No caso de informações mais voláteis, talvez seja mais seguro expurgar os registros depois de alguns segundos ou após um minuto.

Vale a pena mencionar que o método de consulta descrito aqui é conhecido como uma **consulta recursiva**, pois cada servidor que não tiver as informações solicitadas poderá encontrá-las em algum lugar e informar o que encontrou. Também é possível uma forma alternativa. Nessa estratégia, quando não pode ser satisfeita no local, a consulta falha, mas é retornado o nome do próximo servidor a ser consultado ao longo da linha. Alguns servidores não implementam consultas recursivas e sempre retornam o nome do próximo servidor a ser consultado.

Também vale a pena destacar que, quando um cliente DNS deixar de obter uma resposta antes de seu timer expirar, em geral ele tentará acessar outro servidor na próxima vez. A suposição aqui é que provavelmente o servidor está inativo, e não que a solicitação ou a resposta se perdeu.

Embora o DNS seja extremamente importante para o funcionamento correto da Internet, tudo que ele faz na realidade é mapear nomes simbólicos de máquinas em seus endereços IP. Ele não ajuda a localizar pessoas, recursos, serviços ou objetos em geral. Para localizar esses itens, outro serviço de diretório é definido, o chamado **LDAP (Light-weight Directory Access Protocol).** Esse protocolo é uma versão simplificada do serviço de diretório X.500 do OSI e é descrito na RFC 2251. Ele organiza as informações como uma árvore e permite pesquisas em diferentes componentes. O LDAP pode ser considerado um "catálogo telefônico de assinantes". Não o discutiremos mais neste livro; porém, para obter mais informações, consulte (Weltman e Dahbura, 2000).

[T2] 7.2 Correio eletrônico

O **correio eletrônico** — ou **e-mail**, como é chamado por muitos de seus fãs — já existe há mais de duas décadas. Antes de 1990, ele era empregado principalmente nos meios acadêmicos. Durante os anos 90, ficou conhecido para o público em geral e seu uso cresceu exponencialmente, até alcançar um número de mensagens de correio eletrônico enviadas por dia imensamente maior que o número de cartas remetidas pelo **correio convencional** (isto é, cartas escritas em papel).

Como a maioria das outras formas de comunicação, o correio eletrônico tem suas próprias convenções e estilos. Em particular, é muito informal e tem baixo limiar de uso. Pessoas que nunca sonhariam telefonar ou mesmo escrever uma carta para uma pessoa muito importante não hesitam nem um segundo em enviar uma

mensagem de correio eletrônico escrita às pressas e sem cuidado.

O correio eletrônico está repleto de elementos de jargão, como AP (A Propósito), RCTR (Rolando no Chão de Tanto Rir) e EMHO (Em Minha Humilde Opinião). Muitas pessoas também empregam pequenos símbolos ASCII chamados **smileys** ou **emoticons** em suas mensagens de correio eletrônico. Alguns dos mais interessantes estão reproduzidos na Figura 7.6. Na maioria dos casos, basta girar o livro 90 graus à direita para que eles se tornem mais claros. Se quiser conhecer um pequeno livro com mais 650 smileys, consulte (Sanderson e Dougherty, 1993).

[arte: ver original p. 589]

[T]Tabela

Smiley **Significado**

:-) Estou feliz

:-(Estou triste/zangado

:-| Estou indiferente

;-) Estou piscando

:-(O) Estou gritando

:-(*) Estou vomitando

=|:-) Abraham Lincoln

=):-) Tio Sam

*<:-) Papai Noel

<:-) Bobo

(-: Australiano

:-)X Homem com gravata borboleta

:+) Narigudo

:-)) Queixo duplo

:-{) Bigode

#:-) Cabelo embaraçado

8-) Usa óculos

C:-) Cabeção

[F]Figura 7.6

[FL] Alguns smileys (ou emoticons). Eles não cairão no exame final :-)

Os primeiros sistemas de correio eletrônico consistiam simplesmente em protocolos de transferência de arquivos, com a convenção de que a primeira linha de cada mensagem (isto é, o arquivo) deveria conter o endereço do destinatário. À medida que o tempo passou, as limitações dessa estratégia se tornaram mais óbvias. Algumas reclamações eram:

1. Enviar uma mensagem a um grupo de pessoas era inconveniente. Com freqüência, os gerentes precisavam desse recurso para enviar memorandos a todos os seus subordinados.

2. As mensagens não tinham estrutura interna, o que dificultava seu processamento por parte do computador. Por exemplo, se uma mensagem encaminhada estivesse incluída no corpo de outra mensagem, era difícil extrair a parte encaminhada da mensagem recebida.

3. O remetente nunca sabia se uma mensagem havia chegado ou não.

4. Se alguém que pretendesse se ausentar do trabalho por algumas semanas e quisesse deixar sua secretária com a responsabilidade de cuidar da correspondência recebida durante esse tempo, ela teria dificuldades para realizar essa tarefa.

5. A interface do usuário não estava bem integrada ao sistema de transmissão. Portanto, o usuário tinha de editar um arquivo, sair do editor e chamar o programa de transferência de arquivos.

6. Não era possível criar e enviar mensagens contendo uma mistura de textos,

desenhos, fax e voz.

À medida que as pessoas ganharam mais experiência, foram propostos sistemas de correio eletrônico mais elaborados. Em 1982, as propostas relativas a correio eletrônico da ARPANET foram publicadas como a RFC 821 (protocolo de transmissão) e a RFC 822 (formato de mensagens). Revisões menores, publicadas nas RFCs 2821 e 2822, se tornaram padrões da Internet, mas todos ainda se referem ao correio eletrônico da Internet como a RFC 822.

Em 1984, o CCITT esboçou sua recomendação X.400. Após duas décadas de concorrência, os sistemas de correio eletrônico baseados na RFC 822 passaram a ser amplamente usados, enquanto os sistemas baseados na X.400 desapareceram no horizonte. O fato de um sistema criado por alguns alunos da faculdade de ciência da computação ter sobrepujado um padrão internacional aprovado não só por todas as empresas de telecomunicações do mundo inteiro, mas também pelo governo de muitos países e por uma parte substancial do setor de informática nos faz lembrar da história bíblica de Davi e Golias.

O sucesso da RFC 822 não se deve apenas à sua qualidade, mas também ao projeto ruim da X.400, cuja complexidade impedia sua implementação adequada. Tendo de escolher entre um sistema de correio eletrônico baseado na RFC 822 simples mas que funcionava e um sistema baseado na X.400 supostamente maravilhoso mas que não funcionava, muitas empresas optaram pelo primeiro. Talvez possamos tirar uma lição de tudo isso na hora de tomar alguma decisão importante. De qualquer modo, nossa discussão sobre o correio eletrônico se concentrará nos sistemas de correio eletrônico da Internet.

[T3] 7.2.1 Arquitetura e serviços

Nesta seção, apresentaremos uma visão geral do que os sistemas de correio eletrônico podem fazer e como eles estão organizados. Em geral, eles consistem

em dois subsistemas: os **agentes do usuário**, que permitem que as pessoas leiam e enviem mensagens, e os **agentes de transferência de mensagens**, que deslocam as mensagens da origem até o destino. Os agentes do usuário são programas locais que oferecem um método baseado em comandos, baseado em menus ou gráfico para interagir com o sistema de correio eletrônico. Normalmente, os agentes de transferência de mensagens são **daemons** do sistema, ou seja, processos executados no segundo plano. Sua tarefa é mover as mensagens de correio eletrônico pelo sistema.

Em geral, os sistemas de correio eletrônico admitem cinco funções básicas. Vamos examiná-las.

A **composição** se refere ao processo de criar mensagens e respostas. Apesar de qualquer editor de textos poder ser usado para o corpo da mensagem, o sistema em si pode auxiliá-lo com o endereçamento e com os inúmeros campos de cabeçalho associados a cada mensagem. Por exemplo, quando responde a uma mensagem, o sistema de correio eletrônico pode extrair o endereço do remetente da mensagem recebida e incluí-lo automaticamente no lugar adequado da resposta.

A **transferência** se refere ao deslocamento de uma mensagem entre o remetente e o destinatário. Em geral, isso exige o estabelecimento de uma conexão com o destino ou com alguma máquina intermediária, a transmissão de uma mensagem e o encerramento da conexão. O sistema de correio eletrônico pode fazer isso automaticamente, sem perturbar o usuário.

A **geração de relatórios** está relacionada ao fato de informar o remetente sobre o que aconteceu com a mensagem. Ela foi entregue? Foi rejeitada? Perdeu-se? Existem inúmeras aplicações em que a confirmação da entrega da mensagem é importante e pode ter até mesmo significação legal ("Bem, Excelência, meu sistema de correio eletrônico não é confiável. Acho que a intimação eletrônica

que me enviaram se perdeu em algum lugar").

A **exibição** das mensagens recebidas é necessária para que as pessoas possam ler suas mensagens de correio eletrônico. Às vezes, é preciso fazer a conversão ou invocar um visualizador especial, como acontece quando a mensagem é um arquivo PostScript ou de voz digitalizada. Algumas vezes, também há tentativas de conversões e formatações simples.

A **disposição** é a última etapa e se refere ao que o destinatário faz com a mensagem depois de recebê-la. Dentre as possibilidades estão jogá-la fora, gravá-la etc. Também deveria ser possível recuperar e reler as mensagens gravadas, encaminhá-las ou processá-las de outras formas.

Além desses serviços básicos, alguns sistemas de correio eletrônico, em especial os sistemas internos de empresas, oferece uma grande variedade de recursos avançados. Agora vamos mencionar alguns deles resumidamente. Talvez as pessoas queiram que suas mensagens sejam encaminhadas até elas quando se mudarem ou quando ficarem fora durante um bom tempo; portanto, o sistema deverá ser capaz de fazê-lo de forma automática.

A maior parte dos sistemas permite que os usuários criem **caixas de correio** para armazenar a correspondência recebida. São necessários comandos para criar e destruir caixas de correio, inspecionar seu conteúdo, inserir e excluir mensagens das caixas de correio e assim por diante.

Com freqüência, os gerentes de empresas precisam enviar uma mensagem a cada um de seus subordinados, clientes ou fornecedores. Essa necessidade deu origem à idéia de uma **lista de debate** (**mailing list**), uma lista de endereços de correio eletrônico. Quando uma mensagem é enviada a uma lista de debate, cópias idênticas são entregues a cada pessoa incluída na lista.

Outros recursos avançados são as cópias carbono (cc), as cópias carbono cegas (bcc), as mensagens de alta prioridade, as mensagens secretas (isto é,

criptografadas), os destinatários alternativos caso o destinatário principal não esteja disponível no momento, e ainda a possibilidade das secretárias lerem a responderem a correspondência de seus chefes.

Hoje em dia, o correio eletrônico é amplamente utilizado para a comunicação em uma empresa ou entre empresas. Ele permite que funcionários fisicamente distantes (até mesmo em diferentes regiões do planeta) cooperem em projetos de grande complexidade, mesmo que estejam separados por muitos fusos horários. Ao eliminar a maior parte das pistas associadas a classes sociais, faixas etárias e sexo, os debates por meio do correio eletrônico tendem a se concentrar em idéias e não na situação das empresas. Com o correio eletrônico, uma idéia brilhante de um aluno de um curso de verão pode ter um impacto maior do que uma idéia ruim de um vice-presidente executivo.

Uma idéia fundamental em todos os sistemas de correio eletrônico é a distinção entre o **envelope** e seu conteúdo. O envelope encapsula a mensagem. Ele contém todas as informações necessárias para o transporte da mensagem, como endereço de destino, prioridade e nível de segurança, sendo todas elas distintas da mensagem em si. Os agentes de transporte de mensagens utilizam o envelope para executar o roteamento, exatamente como uma agência de correios.

A mensagem dentro do envelope consiste em duas partes: o **cabeçalho** e o **corpo**. O cabeçalho contém informações de controle para os agentes do usuário. O corpo da mensagem diz respeito apenas ao destinatário. Os envelopes e as mensagens são ilustrados na Figura 7.7.

[arte: ver original p. 592]

[Dísticos]

[1] Corpo Cabeçalho Envelope

[2] Envelope

[3] Mensagem

Atenção, produção!

Favor reproduzir os outros dísticos desta figura, pois ficarão em inglês.

[F]Figura 7.7

[FL] Envelopes e mensagens. (a) Correio convencional. (b) Correio eletrônico

[T3] 7.2.2 O agente do usuário

Os sistemas de correio eletrônico têm duas partes básicas, como vimos anteriormente: os agentes do usuário e os agentes de transferência de mensagens. Nesta seção, estudaremos os agentes do usuário. Em geral, um agente do usuário é um programa (às vezes, chamado de leitor de correio eletrônico) que aceita uma variedade de comandos para composição (ou redação), recebimento e resposta a mensagens, bem como para manipular caixas de correio. Alguns agentes do usuário têm uma sofisticada interface baseada em menus ou ícones que exige um mouse, enquanto outros esperam receber comandos de 1 caractere do teclado. Porém, em termos funcionais, essas interfaces são iguais. Alguns sistemas são comandados por meio de ícones ou menus, mas também têm atalhos pelo teclado.

[T4] O envio de mensagens de correio eletrônico

Para que possa enviar uma mensagem de correio eletrônico, o usuário deve incluir nela o endereço de destino e possivelmente alguns outros parâmetros. A mensagem pode ser produzida com um editor de textos independente, com um programa de processamento de textos, ou talvez com um editor de textos especializado, incorporado ao agente do usuário. O endereço de destino deverá estar em um formato com o qual o agente do que o usuário possa lidar. Muitos agentes do usuário esperam endereços no formato *usuário@endereço-dns.* Tendo em vista que já estudamos o DNS em uma seção anterior deste capítulo, não

repetiremos esse assunto aqui.

No entanto, vale a pena observar que existem outras formas de endereçamento. Em particular, os endereços X.400 têm aspecto radicalmente diferente dos endereços DNS. Eles são compostos por pares *atributo* = *valor* separados por barras; por exemplo:

[TD]

/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/ [TN]

Esse endereço especifica um país, um estado, uma localidade, um endereço pessoal e um nome comum (Ken Smith). Existem muitos outros atributos; portanto, é possível enviar uma mensagem de correio eletrônico para alguém cujo endereço exato de correio eletrônico você não conhece, desde que conheça outros atributos em número suficiente (por exemplo, empresa e cargo). Embora os nomes X.400 sejam bem menos convenientes que os nomes DNS, a maioria dos sistemas de correio eletrônico tem nomes alternativos (às vezes chamados apelidos) que permitem aos usuários digitarem ou selecionarem o nome de uma pessoa e obterem o endereço e correio eletrônico apropriado. Conseqüentemente, mesmo com endereços X.400, em geral não é necessário digitar esses strings estranhos.

A maioria dos sistemas de correio eletrônico dispõe de recursos de listas de debate, de modo que um usuário possa enviar a mesma mensagem a uma lista de pessoas com um único comando. Se a lista de debate for mantida no loca, o agente do usuário poderá enviar uma mensagem separada para cada destinatário. No entanto, se a lista for mantida remotamente, as mensagens serão expandidas nesse local remoto. Por exemplo, se um grupo de ornitólogos tiver uma lista de debate com o nome *birders* instalada em *meadowlark.arizona.edu*, qualquer mensagem enviada para *birders@meadowlark.arizona.edu* será roteada para a

University of Arizona e lá será expandida em mensagens individuais para todos os membros da lista de debate, onde quer que eles estejam. Os usuários dessa lista de debate não são capazes de reconhecê-la como uma lista de debate. Ela poderia ser simplesmente a caixa de correio pessoal do professor Gabriel O. Birders.

[T4] Leitura de correio eletrônico

Normalmente, quando é acionado, um agente do usuário verifica a caixa de correio do usuário em busca de mensagens recebidas antes de exibir algo na tela. Em seguida, ele pode anunciar o número de mensagens contidas na caixa de correio ou exibir um resumo de uma linha de cada uma delas e aguardar um comando.

Um exemplo de como um agente do usuário funciona pode ser extraído de um típico cenário de correio eletrônico. Depois de inicializar o agente do usuário, o usuário solicita um resumo de sua correspondência. Em seguida, é mostrada na tela uma janela semelhante à da Figura 7.8. Cada linha se refere a uma mensagem. Nesse exemplo, a caixa de correio contém oito mensagens.

[arte: ver original p. 593]

[T]Tabela

Nº	Flags	Bytes	Sender (Remetente)	Subject (Assunto)
1	K	1030	asw	Alterações no MINX
2	KA	6348	trudy	Comentários sobre o material enviado
3	K F	4519	Amy N. Wong	Solicitação de informações
4		1236	bal	Bioinformática
5		104110	kaashoek	Texto do documento DCS
6		1223	Frank	Re: Você examinaria uma proposta de concessão?
7		3110	guido	Comentários do revisor do artigo

8 1204 dmr Re: Visita dos meus alunos

[F]Figura 7.8

[FL] Um exemplo de exibição do conteúdo de uma caixa de correio

Cada linha da janela contém vários campos extraídos do envelope ou do cabeçalho da mensagem correspondente. Em um sistema de correio eletrônico simples, a escolha dos campos exibidos na tela é definida pelo programa. Em um sistema mais sofisticado, o usuário pode especificar quais campos deverão ser exibidos fornecendo um **perfil de usuário**, um arquivo que descreve o formato de exibição. Nesse exemplo básico, o primeiro campo é o número da mensagem. O segundo, *Flags*, pode conter um *K*, indicando que a mensagem não é nova, mas que foi lida anteriormente e mantida na caixa de correio; um *A*, indicando que a mensagem já foi respondida; e/ou um *F*, indicando que ela foi encaminhada a outra pessoa. Outros flags também são possíveis.

O terceiro campo informa o tamanho da mensagem e o quarto campo informa quem a enviou. Como esse campo foi simplesmente extraído da mensagem, ele pode conter nomes, nomes completos, iniciais, nomes de login ou qualquer outra informação que o remetente queira incluir. Por fim, o campo *Subject* apresenta um breve resumo do que trata a mensagem. As pessoas que não incluem um campo *Subject* em suas mensagens freqüentemente descobrem que as respostas a suas mensagens de correio eletrônico tendem a não receber a prioridade mais alta.

Após a exibição dos cabeçalhos, o usuário poderá executar qualquer uma entre várias ações, como exibir uma menagem, excluir uma mensagem e assim por diante. Os sistemas mais antigos eram baseados em texto e, em geral, usavam comandos de 1 caractere para executar essas tarefas, como T (digitar mensagem), A (responder mensagem), D (excluir mensagem) e F (encaminhar

mensagem). Um argumento especificava a mensagem em questão. Sistemas mais recentes utilizam interfaces gráficas. Normalmente, o usuário seleciona uma mensagem com o mouse e depois dá um clique em um ícone para digitar, responder, excluir ou encaminhar a mensagem.

O correio eletrônico já evoluiu muito desde o tempo em que era simplesmente um método para transferência de arquivos. Por meio de sofisticados agentes do usuário, é possível gerenciar um grande volume de mensagens de correio eletrônico. Para pessoas que recebem e enviam milhares de mensagens a cada ano, essas ferramentas não têm preço.

[T3] 7.2.3 Formatos de mensagens

Agora passaremos da interface do usuário para o formato das mensagens de correio eletrônico propriamente ditas. Primeiro, estudaremos as mensagens básicas em código ASCII utilizando a RFC 822. Depois disso, veremos as extensões de multimídia aplicadas à RFC 822.

[T4] RFC 822

As mensagens consistem em um envelope primitivo (descrito na RFC 821), em alguns campos de cabeçalho, em uma linha em branco e no corpo da mensagem. Cada campo de cabeçalho consiste (logicamente) em uma única linha de texto ASCII contendo o nome do campo, um sinal de dois-pontos e, quase sempre, um valor. A RFC 822 é um padrão antigo e não distingue claramente os campos do envelope dos campos do cabeçalho. Embora ela tenha sido revista na RFC 2822, não foi possível refazê-la completamente, devido a sua utilização difundida. Em uso normal, o agente do usuário cria uma mensagem e a repassa ao agente de transferência de mensagens que, em seguida, emprega alguns dos campos de cabeçalho para criar o envelope, em uma mistura meio antiquada de mensagem e

envelope.

Os principais campos de cabeçalho relacionados ao transporte de mensagens são listados na Figura 7.9. O campo *To*: indica o endereço DNS do destinatário principal. Também é possível ter vários destinatários. O campo *Cc*: contém os endereços dos destinatários secundários (se houver). Em termos de entrega, não há distinção entre os destinatários principal e secundário. Trata-se de uma diferença inteiramente psicológica, importante apenas para as pessoas envolvidas, mas que não afeta o sistema de correio. O termo Cc: (cópia carbono) já está meio ultrapassado, pois os computadores não utilizam papel carbono, mas é uma expressão consagrada pelo uso. O campo *Bcc*: (cópia carbono cega) é semelhante ao campo *Cc*:, exceto pelo fato de essa linha ser eliminada de todas as cópias enviadas aos destinatários principais e secundários. Esse recurso permite que as pessoas enviem cópias a terceiros sem que os destinatários principais e secundários saibam disso.

[arte: ver original p. 595]

[T]Tabela

Cabeçalho	**Significado**
To:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) principal(is)
Cc:	O(s) endereço(s) de correio eletrônico do(s) destinatário(s) secundário(s)
Bcc:	O(s) endereço(s) de correio eletrônico para cópias carbono cegas
From:	A(s) pessoa(s) que criou(aram) a mensagem
Sender:	O endereço de correio eletrônico do remetente
Received:	A linha incluída por cada agente de transferência ao longo da rota
Return-Path:	Pode ser usado para identificar um caminho de volta ao remetente

[F]Figura 7.9

[FL] Os campos do cabeçalho RFC 822 relacionados ao transporte de mensagens

Os dois campos seguintes, *From*: e *Sender*: informam quem escreveu e enviou a mensagem, respectivamente. Nem sempre esses campos conterão valores iguais. Por exemplo, um executivo pode escrever uma mensagem, mas na verdade sua secretária é quem acaba transmitindo a mensagem. Nesse caso, o executivo seria listado no campo *From*: e a secretaria no campo *Sender*:. O campo *From*: é obrigatório, ao passo que *Sender*: pode ser omitido se for igual a *From*:. Esses campos são necessários caso a mensagem não possa ser entregue e tenha de ser devolvida ao remetente.

Uma linha contendo *Received*: é incluída por cada agente de transferência de mensagens ao longo do percurso. A linha contém a identidade do agente, a data e a hora em que a mensagem foi recebida e outras informações que podem ser usadas para localização de bugs no sistema de roteamento.

O campo *Return-Path*: é incluído pelo último agente de transferência de mensagens e seu objetivo é informar como voltar ao remetente. Na teoria, essas informações podem ser obtidas a partir de todos os cabeçalhos *Received*: (exceto pelo nome da caixa de correio do remetente), mas ele raramente é preenchido dessa forma e, em geral, contém apenas o endereço do remetente.

Além dos campos da Figura 7.9, as mensagens da RFC 822 também podem conter uma variedade de campos de cabeçalho utilizados pelos agentes do usuário ou pelos destinatários. Os mais comuns estão listados na Figura 7.10. A maior parte deles é auto-explicativa e assim não entraremos em detalhes sobre todos eles.

Às vezes o campo *Reply-To*: é usado quando nem a pessoa que redigiu a mensagem e nem a pessoa que enviou a mensagem quer ver a resposta. Por exemplo, um gerente de marketing escreve uma mensagem apresentando um novo produto aos clientes. A mensagem é enviada pela secretária, mas o campo

Reply-To: lista o chefe do departamento de vendas, que pode responder às perguntas e receber pedidos do produto. Esse campo também é útil quando o remetente tem duas contas de correio eletrônico e deseja que a resposta vá para a outra conta.

[arte: ver original p. 596]

[T]Tabela

Cabeçalho	**Significado**
Date:	A data e a hora em que a mensagem foi enviada
Reply-To:	O endereço de correio eletrônico para onde as respostas devem ser enviadas
Message-Id:	O número exclusivo que será usado para fazer referência a essa mensagem posteriormente
In-Reply-To:	Message-Id da mensagem original correspondente a essa reposta
References:	Outras Message-Ids relevantes
Keyword	Palavras-chaves do usuário
Subject:	Pequeno resumo da mensagem apresentado em apenas uma linha

[F]Figura 7.10

[FL] Alguns campos usados no cabeçalho de mensagem RFC 822

O documento RFC 822 menciona explicitamente que os usuários têm permissão de criar novos cabeçalhos para seu próprio uso, desde que esses cabeçalhos comecem com o string *X-*. É certo que nenhum cabeçalho utilizará nomes começando com *X-* no futuro, a fim de evitar conflitos entre cabeçalhos oficiais e particulares. Às vezes, alguns estudantes pretensiosos criam campos como *X-Fruta-do-Dia*: ou *X-Doença-da-Semana*:, que são válidos, mas nem sempre muito esclarecedores.

Depois dos cabeçalhos, vem o corpo da mensagem. Os usuários podem incluir o que quiserem. Algumas pessoas encerram suas mensagens com assinaturas elaboradas, incluindo desenhos simples em código ASCII, citações de autores ou personalidades, declarações políticas e ressalvas de todos os tipos (por exemplo, a ABC Corporation não é responsável por minhas opiniões; ela não é nem mesmo capaz de compreendê-las).

[T4] MIME — Multipurpose Internet Mail Extensions

Nos primórdios da ARPANET, o correio eletrônico consistia exclusivamente em mensagens de texto escritas em linguagem comum e expressas em código ASCII. Para esse ambiente, a RFC 822 fez tudo o que era preciso: especificou os cabeçalhos, mas deixou o conteúdo inteiramente a cargo dos usuários. Hoje em dia, na Internet mundial, essa estratégia já deixou de ser adequada. Os problemas incluem o envio e o recebimento de:

1. Mensagens em idiomas com acentos (por exemplo, português e alemão).
2. Mensagens em alfabetos não latinos (por exemplo, hebraico e russo).
3. Mensagens em idiomas sem alfabetos (por exemplo, chinês e japonês).
4. Mensagens que não contêm textos (por exemplo, áudio ou imagens).

Uma solução foi proposta na RFC 1341 e atualizada nas RFCs 2045 a 2049. Essa solução, denominada **MIME (Multipurpose Internet Mail Extensions)** está sendo amplamente utilizada. Agora vamos descrevê-la. Para obter informações adicionais sobre o MIME, consulte as RFCs.

A idéia básica do MIME é continuar a usar o formato RFC 822, mas incluir uma estrutura para o corpo da mensagem e definir regras para mensagens que não utilizam o código ASCII. Por manterem o formato 822, as mensagens no padrão MIME podem ser enviadas através da utilização dos protocolos e programas de correio eletrônico existentes no mercado. Só é necessário alterar os programas de

envio e recebimento, o que os próprios usuários podem fazer.

O MIME define cinco novos cabeçalhos de mensagens, como mostra a Figura 7.11. O primeiro deles informa ao agente do usuário receptor da mensagem não apenas que ele está lidando com uma mensagem MIME, como também a versão do padrão MIME que está sendo usada. Presume-se que qualquer mensagem que não contenha um cabeçalho *MIME-Version*: seja uma mensagem de texto simples escrita em linguagem comum e processada como tal.

[arte: ver original p. 597]

[T]Tabela

Cabeçalho	**Significado**
MIME-Version:	Identifica a versão do MIME
Content-Description:	String inteligível que identifica o conteúdo da mensagem
Content-Id:	Identificador exclusivo
Content-Transfer-Encoding:	Como o corpo da mensagem é codificado para transmissão
Content-Type:	Tipo e formato do conteúdo

[F]Figura 7.11

[FL] Os cabeçalhos RFC 822 incluídos pelo MIME

O cabeçalho *Content-Description*: é um string ASCII que informa o conteúdo da mensagem. Esse cabeçalho é necessário para que o destinatário saiba se vale a pena decodificar e ler a mensagem. Se o string informar "Foto do ratinho da Bárbara" e a pessoa que receber a correspondência não for muito chegada a esses bichinhos, provavelmente a mensagem será descartada em vez de ser decodificada em uma fotografia colorida de alta resolução.

O cabeçalho *Content-Id* identifica o conteúdo. Ele utiliza o mesmo formato do

cabeçalho *Message-Id*: padrão.

Content-Transfer-Encoding: informa como o corpo da mensagem está codificado para transmissão através de uma rede que possa fazer alguma objeção a caracteres que não sejam letras, números e outros sinais de pontuação. São fornecidos cinco esquemas (e uma saída para novos esquemas). O esquema mais simples é formado apenas por texto em código ASCII. Os caracteres ASCII utilizam 7 bits e podem ser transportados diretamente pelo protocolo de correio eletrônico, desde que nenhuma linha ultrapasse 1000 caracteres.

O esquema mais simples seguinte é igual ao anterior, mas utiliza caracteres de oito bits, isto é, todos os valores de 0 a 255, inclusive. Esse esquema de codificação viola o protocolo de correio eletrônico (original) da Internet, mas é usado por algumas partes da Internet que implementam extensões ao protocolo original. Embora declarar a codificação não a torne obrigatoriamente válida, torná-la explícita poderá ao menos explicar o que aconteceu quando algo der errado. As mensagens que utilizam a codificação de 8 bits também devem aderir ao tamanho máximo de linha padrão.

Pior ainda são as mensagens que utilizam a codificação binária. Essas mensagens são arquivos binários que não só utilizam todos os 8 bits, mas também não respeitam sequer o limite de linha de 1000 caracteres. Os programas executáveis estão nessa categoria. Não há garantia de que as mensagens em código binário cheguem corretamente a seu destino, mas muita gente as envia assim mesmo.

A maneira correta de codificar mensagens binárias é utilizar a **codificação base64**, às vezes chamada **armadura ASCII**. Nesse esquema, grupos de 24 bits são divididos em até quatro unidades de 6 bits, com cada unidade sendo enviada como um caractere ASCII válido. A codificação é "A" para 0, "B" para 1 e assim por diante, seguida pelas 26 letras minúsculas, pelos dez dígitos e finalmente + e / para 62 e 63, respectivamente. As seqüências == e = são usadas para indicar

que o último grupo continha apenas 8 ou 16 bits, respectivamente. Os caracteres de retorno de cursor e avanço de linha são ignorados; portanto, podem ser inseridos à vontade para manter as linhas curtas. Utilizando-se esse esquema, é possível enviar textos binários com total segurança.

Para as mensagens quase totalmente em ASCII, mas com alguns caracteres não pertencentes ao código ASCII, a codificação base64 se mostra um tanto ineficiente. Em vez disso, é utilizado um método conhecido como **codificação quoted-printable.** Trata-se apenas de um código ASCII de 7 bits, com todos os caracteres acima de 127 codificados como um sinal de igualdade seguido pelo valor do caractere como dois dígitos hexadecimais.

Em resumo, os dados binários deverão ser enviados codificados no formato base64 ou quoted-printable. Quando houver motivos válidos para não utilizar um desses esquemas, será possível especificar uma decodificação definida pelo usuário no cabeçalho *Content-Transfer-Encoding*:.

O último cabeçalho mostrado na Figura 7.11 é na realidade o mais interessante. Ele especifica a natureza do corpo da mensagem. São definidos sete tipos na RFC 1521, e cada um deles tem um ou mais subtipos. O tipo e o subtipo são separados por uma barra, como em:

[TD] Content-Type: video/mpeg [TN]

O subtipo deve ser informado explicitamente no cabeçalho; não são fornecidos valores predefinidos. A lista inicial de tipos e subtipos especificados na RFC 2045 é mostrada na Figura 7.12. Muitos tipos e subtipos novos foram incluídos desde então, e outras entradas são incluídas sempre que necessário.

[arte: ver original p. 599]

[T]Tabela

Tipo	**Subtipo**	**Descrição**
Text	Plain	Texto sem formatação

	Enriched	Texto incluindo comandos simples de formatação
Image	Gif	Fotografia no formato GIF
	Jpeg	Fotografia no formato JPEG
Audio	Basic	Som audível
Video	Mpeg	Filme no formato MPEG
Application	Octet-stream	Uma seqüência de bytes não interpretada
	PostScript	Um documento que pode ser impresso em PostScript
	Rfc822	Uma mensagem no formato MIME RFC 822
Message	Partial	A mensagem foi dividida para transmissão
	External-body	A mensagem deve ser obtida através da rede
	Mixed	Partes independentes na ordem especificada
Multipart	Alternative	A mesma mensagem em diferentes formatos
	Parallel	As partes devem ser vistas simultaneamente
	Digest	Cada parte é uma mensagem RFC 822 completa

[F]Figura 7.12

[FL] Os tipos e subtipos MIME definidos na RFC 2045

Agora vamos percorrer a lista de tipos. O tipo *text* se destina a texto ASCII simples. A combinação *text/plain* se destina a mensagens comuns que podem ser exibidas na forma como são recebidas, sem necessidade de codificação e de maior processamento. Essa opção permite que mensagens comuns sejam transportadas no MIME com apenas alguns cabeçalhos extras.

O subtipo *text/enriched* permite que uma linguagem de marcação simples seja incluída no texto. Essa linguagem oferece uma forma independente do sistema para expressar negrito, itálico, tamanhos em pontos menores e maiores, recuos, justificação, caracteres subscritos e sobrescritos, além de um layout de página simples. A linguagem de marcação se baseia na SGML (Standard Generalized

Markup Language), também usada como base para a HTML da World Wide Web.

Por exemplo, a mensagem:

[TD]

A <italic> morsa </italic> disse: chegou o <bold> momento </bold> de ... [TN]

seria exibida como

A *morsa* disse: chegou o **momento** de ...

Cabe ao sistema receptor escolher a interpretação adequada. Se estiverem disponíveis, os atributos negrito e itálico poderão ser usados; caso contrário, será possível utilizar cores, piscamento, sublinhado, vídeo reverso etc. para dar ênfase. Diferentes sistemas podem escolher e escolhem opções distintas.

Quando a Web se tornou popular, foi acrescentado um novo subtipo *text/html* (na RFC 2854) para permitir o envio de páginas da Web em correio eletrônico RFC 822. Um subtipo correspondente à linguagem de marcação extensível, *text/xml*, é definido na RFC 3023. Estudaremos a HTML e a XML mais adiante neste capítulo.

O próximo tipo MIME é *image*, usado para transmitir fotografias. Atualmente, são utilizados muitos formatos para armazenar e transmitir imagens, com e sem compactação. Dois deles, GIF e JPEG, estão incorporados em quase todos os navegadores, mas também existem muitos outros que foram acrescentados à lista original.

Os tipos *audio* e *video* se destinam a som e imagens em movimento, respectivamente. Observe que a opção *video* inclui apenas as informações visuais, e não a trilha sonora. Se um filme com som tiver de ser transmitido, as porções de vídeo e áudio talvez tenham de ser transmitidas separadamente, dependendo do sistema de codificação utilizado. O primeiro formato de vídeo definido foi criado pelo grupo Moving Pictures Experts Group (MPEG), mas desde então foram acrescentados outros formatos. Além de *audio/basic*, foi incluído na RFC 3003

um novo tipo de áudio, *audio/mpeg*, para permitir que as pessoas enviem arquivos de áudio MP3 por correio eletrônico.

O tipo *application* é um depósito para formatos que exigem um tipo de processamento externo, não cobertos por um dos outros tipos. Um *octet-stream* é apenas uma seqüência de bytes não interpretados. Ao receber um fluxo como esse, um agente do usuário provavelmente irá exibi-lo, sugerindo ao usuário que esse fluxo seja copiado em um arquivo e solicitando um nome de arquivo. O processamento subseqüente caberá então ao usuário.

O outro subtipo definido é *postscript*, que se refere à linguagem PostScript definida pela Adobe Systems e amplamente utilizada para descrição de páginas impressas. Muitas impressoras têm interpretadores PostScript embutidos. Apesar de um agente do usuário poder simplesmente chamar um interpretador PostScript externo para exibir os arquivos PostScript recebidos, podem ocorrer problemas nessa operação. PostScript é uma linguagem de programação muito bem desenvolvida. Se tivesse tempo para isso, uma pessoa suficientemente maso-quista poderia desenvolver um compilador C ou um sistema de gerenciamento de bancos de dados em PostScript. A exibição de uma mensagem PostScript recebida é realizada executando-se o programa PostScript que ela contém. Além de exibir texto, esse programa pode ler, modificar ou eliminar os arquivos do usuário, além de produzir outros efeitos colaterais terríveis.

O tipo *message* permite que uma mensagem seja totalmente encapsulada em outra. Esse esquema é útil para, por exemplo, encaminhar mensagens de correio eletrônico. Quando uma mensagem RFC 822 completa é encapsulada em uma mensagem externa, deve-se utilizar o subtipo *rfc822*.

O subtipo *partial* permite a divisão de uma mensagem encapsulada e enviar essas partes separadas (por exemplo, se a mensagem encapsulada for longa demais). Os parâmetros tornam possível montar novamente todas as partes em ordem

correta no destino.

Por fim, o subtipo *external-body* pode ser usado para mensagens muito longas (por exemplo, filmes de vídeo). Em vez de incluir o arquivo MPEG na mensagem, um endereço FTP é fornecido, e o agente do usuário do receptor pode obtê-lo pela rede no momento em que for necessário. Esse recurso é especialmente útil durante o envio de um filme a uma lista de debate, e espera-se que apenas alguns deles assistam ao filme (pense na correspondência eletrônica não desejada contendo vídeos de publicidade).

O último tipo é *multipart*, que permite a uma mensagem conter mais de uma parte, com o início e o fim de cada parte claramente delimitados. O subtipo *mixed* permite que cada parte seja diferente, sem a imposição de uma estrutura adicional. Muitos programas de correio eletrônico permitem ao usuário incluir um ou mais anexos em uma mensagem de texto. Esses anexos são enviados usando-se o tipo *multipart*.

Em contraste com o tipo *multipart*, o subtipo *alternative* permite que a mesma mensagem seja incluída várias vezes, mas expressa em dois ou mais meios diferentes. Por exemplo, uma mensagem poderia ser enviada em código ASCII simples, em RTF e em PostScript. Um agente do usuário projetado de forma adequada que recebesse essa mensagem poderia exibi-la em PostScript, se possível. A segunda opção seria RTF. Se nenhuma dessas opções fosse possível, o arquivo ASCII puro seria exibido. As partes seriam ordenadas da mais simples para a mais complexa, para que os destinatários com agentes do usuário anteriores ao lançamento do MIME pudessem compreender a mensagem (por exemplo, até mesmo um usuário com sistema anterior ao MIME poderia ler um texto ASCII simples).

O subtipo *alternative* também pode ser usado para várias linguagens de programação. Nesse contexto, a Pedra de Roseta pode ser considerada uma

antiga mensagem *multipart/alternative*.

Um exemplo de multimídia é mostrado na Figura 7.13. Aqui, uma saudação de aniversário é transmitida tanto como texto quanto como uma canção. Se o destinatário tiver recursos de áudio, o agente do usuário obterá o arquivo de som, *birthday.snd*, e poderá reproduzi-lo. Caso contrário, a letra da música será exibida na tela em silêncio. As partes são delimitadas por dois hifens seguidos do string (gerado por software) especificado no parâmetro *boundary*.

[arte: ver original da p. 601]

[TD]

From: elinor@abcd.com

To: carolyn@xyz.com

MIME-Version: 1.0

Message-Id: <0704760941.AA00747@abcd.com>

Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm

Subject: Earth orbits sun integral numbers of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm

Content-Type: text/enriched

Happy birthday to you

Happy birthday to you

Happy birthday dear <bold> Carolyn </bold>

Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm

Content-Type: message/external-body;

access-type="anon-ftp";

site="bicycle.abcd.com";

directory="pub";

name="birthday.snd"

content-type: audio/basic

content-transfer-encoding: base64

--qwertyuiopasdfghjklzxcvbnm -- [TN]

[F]Figura 7.13

[FL] Uma mensagem em várias partes contendo as alternativas de texto formatado e áudio

Observe que o cabeçalho *Content-Type* ocorre em três posições nesse exemplo. No nível mais alto, ele indica que a mensagem tem várias partes. Dentro de cada parte, são indicados seu tipo e subtipo. Por fim, dentro do corpo da segunda parte, esse cabeçalho é necessário para informar ao agente do usuário que tipo de arquivo externo será obtido. Para indicar essa pequena diferença em termos de uso, empregamos letras minúsculas, apesar de todos os cabeçalhos não fazerem diferença entre letras maiúsculas e minúsculas.

O cabeçalho *content-transfer-encoding* também é necessário para qualquer corpo externo que não esteja codificado como ASCII de 7 bits.

Voltando aos subtipos de mensagens de várias partes, existem mais duas possibilidades. O subtipo *parallel* é usado quando todas as partes devem ser "vistas" simultaneamente. Por exemplo, com freqüência os filmes têm um canal de áudio e um canal de vídeo. Os filmes serão mais efetivos se esses dois canais forem reproduzidos em paralelo, e não sucessivamente.

Por fim, o subtipo *digest* é usado quando muitas mensagens são compactadas em

uma mensagem composta. Por exemplo, alguns grupos de discussão da Internet colecionam mensagens de assinantes e as enviam como uma única mensagem *multipart/digest*.

[T3] 7.2.4 Transferência de mensagens

O sistema de transferência de mensagens tem como objetivo transmitir mensagens do remetente ao destinatário. A maneira mais simples de fazer isso é estabelecer uma conexão de transporte entre a máquina de origem e a de destino e, em seguida, transferir a mensagem. Após estudarmos como isso é feito normalmente, examinaremos algumas situações nas quais essa opção não funciona, mostrando o que pode ser feito para contornar o problema.

[T4] SMTP — Simple Mail Transfer Protocol

Dentro da Internet, as mensagens de correio eletrônico são entregues quando a máquina de origem estabelece uma conexão TCP com a porta 25 da máquina de destino. Um daemon de correio eletrônico, que se comunica em **SMTP (Simple Mail Transfer Protocol)** permanece na escuta nessa porta. Esse daemon aceita as conexões recebidas e copia as mensagens que elas contêm para as caixas de correio apropriadas. Se uma mensagem não puder ser entregue, um relatório de erros contendo a primeira parte da mensagem não entregue será retornado ao remetente.

O SMTP é um protocolo ASCII muito simples. Após estabelecer a conexão TCP com a porta 25, a máquina de transmissão, operando como cliente, espera que a máquina de recepção, operando como servidor, comunique-se primeiro. O servidor começa enviando uma linha de texto que fornece sua identidade e informa que está preparado para receber mensagens. Caso não esteja, o cliente encerrará a conexão e tentará outra vez mais tarde.

Se o servidor estiver disposto a receber mensagens, o cliente anunciará de quem veio a mensagem e para quem ela está indo. Se esse receptor existir no local de destino, o servidor dará ao cliente o sinal para enviar a mensagem. Em seguida, o cliente enviará a mensagem e o servidor a confirmará. Não são necessários totais de verificação, porque o TCP fornece um fluxo de bytes confiável. Se houver mais mensagens, elas serão enviadas. Quando todas as mensagens tiverem sido trocadas em ambos os sentidos, a conexão será encerrada. Um exemplo do diálogo necessário para o envio da mensagem ilustrada na Figura 7.13, incluindo os códigos numéricos utilizados pelo SMTP, é mostrado na Figura 7.14. As linhas enviadas pelo cliente são marcadas como *C:*. As linhas enviadas pelo servidor são marcadas como *S:*.

Alguns comentários sobre a Figura 7.14 talvez sejam interessantes. O primeiro comando enviado pelo cliente é *HELO*. Dentre as várias abreviaturas de quatro caracteres para *HELLO*, essa tem muito mais vantagens do que sua concorrente. A razão para que todos os comandos tivessem quatro caracteres se perdeu no tempo.

Na Figura 7.14, a mensagem é enviada para um único destinatário; portanto, apenas um comando *RCPT* é usado. Esses comandos são usados para enviar uma única mensagem a vários destinatários. Cada um deles é confirmado ou rejeitado individualmente. Ainda que alguns destinatários sejam rejeitados (por não existirem no destino), a mensagem poderá ser enviada aos outros.

Por fim, apesar da sintaxe dos comandos de quatro caracteres enviados pelo cliente ser especificada de forma bastante rígida, a sintaxe das respostas é mais flexível. Apenas o código numérico é importante. Cada implementação pode incluir um string qualquer depois do código.

Para entender melhor como o SMTP e alguns dos outros protocolos descritos neste capítulo funcionam, experimente-os. Em todos os casos, vá primeiro até

um equipamento conectado à Internet. Em um sistema UNIX, digite em um shell:

[TD] telnet mail.isp.com 25 [TN]

substituindo *mail.isp.com* pelo nome DNS do servidor de correio do seu provedor. Em um sistema Windows, clique em Iniciar, depois em Executar e, em seguida, digite o comando na caixa de diálogo. Esse comando estabelecerá uma conexão telnet (isto é, TCP) para a porta 25 nessa máquina. A porta 25 é a porta SMTP (veja na Figura 6.27 algumas portas comuns). Provavelmente, você obterá uma resposta semelhante a esta:

[TD]

Trying 192.30.200.66...

Connected to mail.isp.com

Escape character is '^]'.

220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200 [TN]

As três primeiras linhas são do telnet, informando-lhe o que está fazendo. A última linha é do servidor SMTP na máquina remota, anunciando sua disposição para se comunicar com você e aceitar mensagens de correio eletrônico. Para descobrir que comandos ele aceita, digite:

[TD] HELP [TN]

Desse ponto em diante, é possível uma seqüência de comandos como a da Figura 7.14, começando com o comando *HELO* do cliente.

Vale a pena notar que o uso das linhas de texto ASCII para comandos não é um acidente. A maioria dos protocolos da Internet funciona desse modo. O uso de texto ASCII torna os protocolos fáceis de testar e depurar. Eles podem ser testados enviando-se comandos manualmente, como vimos antes, e os dumps das mensagens são fáceis de ler.

[arte: ver original da p. 604]

[TD]

S: 220 xyz.com SMTP service ready

C: HELO abcd.com

S: 250 xyz.com says hello to abcd.com

C: MAIL FROM: <elinor@abcd.com>

S: 250 sender ok

C: RCPT TO: <carolyn@xyz.com>

S: 250 recipient ok

C: DATA

S: 354 Send Mail; end with "." on a line by itself

C: From: elinor@abcd.com

C: To: carolyn@xyz.com

C: MIME-Version: 1.0

C: Message-Id: <0704760941.AA00747@abcd.com>

C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm

C: Subject: Earth orbits sun integral numbers of times

C:

C: This is the preamble. The user agent ignores it. Have a nice day.

C:

C: --qwertyuiopasdfghjklzxcvbnm

C: Content-Type: text/enriched

C: Happy birthday to you

C: Happy birthday to you

C: Happy birthday dear <bold> Carolyn </bold>

C: Happy birthday to you

C:

C: --qwertyuiopasdfghjklzxcvbnm

C: Content-Type: message/external-body;

C: access-type="anon-ftp";

C: site="bicycle.abcd.com";

C: directory="pub";

C: name="birthday.snd"

C:

C: content-type: audio/basic

C: content-transfer-encoding: base64

C: --qwertyuiopasdfghjklzxcvbnm --

C: .

S: 250 message accepted

C: QUIT

S: 221 xyz.com closing connection [TN]

[F]Figura 7.14

[FL] Transferência de uma mensagem de *elinor@abcd.com* para *carolyn@xyz.com*

Apesar do protocolo SMTP ser muito bem definido, ainda podem surgir alguns problemas. Um problema diz respeito ao tamanho das mensagens. Algumas implementações mais antigas não são capazes de lidar com mensagens maiores que 64 KB. Outro problema é causado pelos timeouts. Se o cliente e o servidor tiverem diferentes timeouts, um deles poderá desistir enquanto o outro ainda estiver ocupado, encerrando a conexão inesperadamente. Por fim, em raras situações, podem ocorrer ser iniciadas enxurradas infinitas de mensagens. Por exemplo, se o host 1 estiver enviando uma mensagem para a lista de debate *A* e o host 2 estiver se comunicando com a lista de debate *B* e se cada lista contiver uma entrada para a outra, todas as mensagens enviadas para uma das listas poderá gerar um volume infindável de tráfego de correio eletrônico, a menos que alguém corrija o problema.

Para contornar alguns desses problemas, o STMP estendido (**ESMTP**) foi definido na RFC 2821. Os clientes que quiserem usá-lo deverão enviar inicialmente uma mensagem *EHLO*, em vez de *HELO*. Se essa mensagem for rejeitada, isso significa que o servidor é um servidor STMP comum, e o cliente deverá proceder da forma habitual. Se *EHLO* for aceita, novos comandos e parâmetros serão permitidos.

[T3] 7.2.5 Entrega final

Até agora, supomos que todos os usuários têm máquinas capazes de enviar e receber mensagens de correio eletrônico. Como vimos, o correio eletrônico é entregue fazendo-se o remetente estabelecer uma conexão TCP para o destinatário, e depois transmitir a mensagem por ela. Esse modelo funcionou bem durante décadas, quando todos os hosts da ARPANET (e mais tarde da Internet) estavam de fato on-line o tempo todo para aceitar conexões TCP. Entretanto, com o advento de pessoas que acessam a Internet chamando seu ISP por modem, o modelo falha. O problema é este: o que acontece quando Elinor quer enviar a Carolyn uma mensagem de correio eletrônico e Carolyn não está on-line no momento? Elinor não pode estabelecer uma conexão TCP para Carolyn e, desse modo, não pode executar o protocolo SMTP.

Uma solução é fazer um agente de transferência de mensagens em uma máquina do ISP aceitar correio eletrônico para seus clientes e armazená-lo nas respectivas caixas de correio, na máquina do ISP. Tendo em vista que esse agente pode estar on-line o tempo todo, as mensagens de correio eletrônico podem ser enviadas para ele 24 horas por dia.

[T4] POP3

Infelizmente, essa solução cria outro problema: como o usuário conseguirá o correio eletrônico do agente de transferência de mensagens do ISP? A solução

para esse problema é criar outro protocolo que permita aos agentes de transferência do usuário (em PCs clientes) entrar em contato com o agente de transferência de mensagens (na máquina do ISP) e permitir que as mensagens de correio eletrônico sejam copiadas do ISP para o usuário. Um protocolo desse tipo é o **POP3 (Post Office Protocol Version 3)**, descrito na RFC 1939.

A situação anterior (na qual o transmissor e o receptor têm uma conexão permanente para a Internet) é ilustrada na Figura 7.15(a). Uma situação em que o transmissor está (atualmente) on-line, mas o receptor não está, é ilustrada na Figura 7.15(b).

[arte: ver original p. 606]

[Dísticos]

[1] SMTP Internet Agente de transferência de mensagens Agente do usuário

(a) Host de transmissão Conexão permanente Caixa de correio Host de recepção

[2] SMTP Internet Agente de transferência de mensagens

(b) Host de transmissão Caixa de correio Máquina do ISP

[3] Servidor POP3 POP3 Agente do usuário

Conexão dial-up PC do usuário

[F]Figura 7.15

[FL] (a) Envio e leitura de correio eletrônico quando o receptor tem uma conexão permanente da Internet e o agente do usuário funciona na mesma máquina em que está o agente de transferência de mensagens. (b) Leitura de correio eletrônico quando o receptor tem uma conexão dial-up para um ISP

O POP3 começa quando o usuário inicia o leitor de correio. O leitor de correio chama o ISP (a menos que já exista uma conexão) e estabelece uma conexão TCP

com o agente de transferência de mensagens na porta 110. Depois que a conexão é estabelecida, o protocolo POP3 passa por três estados em seqüência:

1. Autorização.

2. Transações.

3. Atualização.

O estado de autorização lida com o login do usuário. O estado de transação lida com a coleta de mensagens de correio eletrônico do usuário e com a marcação das mensagens para exclusão da caixa de correio. Na realidade, o estado de atualização faz a mensagens de correio eletrônico serem excluídas.

Esse comportamento pode ser observado, digitando-se algo como:

[TD] telnet mail.isp.com 110 [TN]

onde *mail.isp.com* representa o nome DNS do servidor de correio do seu ISP. O telnet estabelece uma conexão TCP para a porta 110, na qual o servidor POP3 fica na escuta. Ao aceitar a conexão TCP, o servidor envia uma mensagem ASCII anunciando sua presença. Normalmente, ele começa com [TD]+0K[TN] seguido por um comentário. Um exemplo de cenário é mostrado na Figura 7.16, começando logo após a conexão TCP ter sido estabelecida. Como antes, as linhas marcadas com *C:* são do cliente (usuário) e as linhas marcadas com *S:* são do servidor (o agente de transferência de mensagens na máquina do ISP).

Durante o estado de autorização, o cliente transmite seu nome de usuário, e depois sua senha. Após um login bem-sucedido, o cliente pode então enviar o comando *LIST*, que faz o servidor listar o conteúdo da caixa de correio, uma mensagem por linha, fornecendo o tamanho dessa mensagem. A lista é encerrada por um ponto.

[arte: ver original da p. 607]

[TD]

S: +0K POP3 server ready

C: USER carolyn

S: +OK

C: PASS vegetables

S: +OK login successful

C: LIST

S: 1 2505

S: 2 14302

S: 3 8122

S:.

C: RETR 1

S: (envia mensagem 1)

C: DELE 1

C: RETR 2

S: (envia mensagem 2)

C: DELE 2

C: RETR 3

S: (envia mensagem 3)

C: DELE 3

C: QUIT

S: +OK POP3 server disconnecting [TN]

[F]Figura 7.16

[FL] Usando POP3 para buscar três mensagens

Em seguida, o cliente pode recuperar mensagens usando o comando *RETR* e marcá-las para exclusão com *DELE*. Quando todas as mensagens estiverem recuperadas (e, possivelmente, marcadas para exclusão), o cliente enviará o comando *QUIT* para encerrar o estado de transação e entrar no estado de

atualização. Quando o servidor excluir todas as mensagens, ele enviará uma resposta e interromperá a conexão TCP.

Embora seja verdade que o protocolo POP3 admite a possibilidade de baixar uma mensagem específica ou um conjunto de mensagens e deixá-las no servidor, a maioria dos programas de correio eletrônico simplesmente baixa tudo e esvazia a caixa de correio. Esse comportamento significa que, na prática, a única cópia está no disco rígido do usuário. Se ele sofrer uma pane, todas as mensagens de correio eletrônico poderão se perder permanentemente.

Agora, vamos resumir rapidamente como o correio eletrônico funciona para os clientes de ISPs. Elinor cria uma mensagem para Carolyn usando algum programa de correio eletrônico (isto é, o agente do usuário) e dá um clique em um ícone para enviá-la. O programa de correio eletrônico entrega a mensagem ao agente de transferência de mensagens no host de Elinor. O agente de transferência de mensagens verifica que ela está endereçada a *carolyn@xyz.com*, e assim utiliza o DNS para pesquisar o registra *MX* correspondente a *xyz.com* (onde *xyz.com* é o ISP de Carolyn). Essa consulta retorna um nome DNS do servidor de correio de *xvz.com*. Agora, o agente de transferência de mensagens procura o endereço IP dessa máquina usando novamente o DNS, por exemplo, empregando *gethostbyname*. Em seguida, ele estabelece uma conexão TCP para o servidor SMTP na porta 25 dessa máquina. Usando uma seqüência de comandos SMTP análoga à da Figura 7.14, ele transfere a mensagem para a caixa de correio de Carolyn e interrompe a conexão TCP.

No devido tempo, Carolyn inicializa seu PC, conecta-se ao ISP e inicia seu programa de correio eletrônico. O programa de correio eletrônico estabelece uma conexão TCP para o servidor POP3 na porta 110 da máquina do servidor de correio do ISP. Em geral, o nome DNS ou o endereço IP dessa máquina é configurado quando o programa de correio eletrônico é instalado ou quando é

feita a inscrição no ISP. Depois que a conexão TCP é estabelecida, o programa de correio eletrônico de Carolyn executa o protocolo POP3 para buscar o conteúdo de caixa de correio para o disco rígido de Carolyn, usando comandos semelhantes aos da Figura 7.16. Uma vez que todo o correio eletrônico é transferido, a conexão TCP é encerrada. De fato, a conexão para o ISP também pode ser interrompida nesse momento, pois toda a correspondência está no disco rígido de Carolyn. É claro que para enviar uma resposta, será preciso usar novamente o ISP; assim, em geral a conexão não é interrompida logo após a busca das mensagens de correio eletrônico.

[T4] IMAP

Para um usuário uma conta de correio eletrônico em um ISP que é sempre acessado a partir de um único PC, o POP3 funciona muito bem e é amplamente usado devido à sua simplicidade e robustez. Porém, é um clichê da indústria de informática que, tão logo algo funciona bem, alguém começa a exigir mais recursos (e a receber mais bugs). Isso também aconteceu com o correio eletrônico. Por exemplo, muitas pessoas têm uma única conta de correio eletrônico no trabalho ou na escola e querem acessá-la no trabalho, em seu PC de casa, no laptop quando estão viajando a negócios e em cybercafés quando estão em férias. Embora o POP3 torne isso possível, pois normalmente ele baixa todas as mensagens armazenadas em cada contato, o resultado é que o correio eletrônico do usuário logo fica espalhado por várias máquinas, mais ou menos ao acaso, e algumas delas nem mesmo pertencem ao usuário.

Essa desvantagem deu origem a um protocolo alternativo de entrega final, o **IMAP (Internet Message Access Protocol)**, definido na RFC 2060. Diferente do POP3, que basicamente supõe que o usuário limpará a caixa de correio em cada contato e trabalhará off-line depois disso, o IMAP pressupõe que todas as mensagens de

correio eletrônico permanecerão no servidor indefinidamente, em várias caixas de correio. O IMAP fornece mecanismos extensos para leitura de mensagens ou mesmo partes de mensagens, um recurso útil quando se utiliza um modem lento para ler a parte de texto de uma mensagem de várias partes com grandes anexos de áudio e vídeo. Tendo em vista que a suposição funcional é que as mensagens não serão transferidas para o computador do usuário com a finalidade de armazenamento permanente, o IMAP fornece mecanismos para criar, destruir e manipular várias caixas de correio no servidor. Desse modo, um usuário pode manter uma caixa de correio para cada correspondente e mover as mensagens da caixa de entrada para essas caixas depois que elas forem lidas.

O IMAP tem muitos recursos, como a possibilidade de endereçar a correspondência não pelo número de chegada, como é feito na Figura 7.8, mas sim usando atributos (por exemplo, forneça a primeira mensagem de Bobbie). Diferente do POP3, o IMAP também pode aceitar mensagens de saída para transporte até o destino, além de entregar a correspondência eletrônica de entrada.

O estilo geral do protocolo IMAP é semelhante ao do POP3, como mostra a Figura 7.16, exceto pelo fato de existirem dezenas de comandos. O servidor IMAP escuta na porta 143. Uma comparação entre POP3 e IMAP é apresentada na Figura 7.17. Porém, devemos observar que nem todo ISP admite os dois protocolos e nem todo programa de correio os aceita. Desse modo, ao escolher um programa de correio eletrônico, é importante descobrir qual(is) protocolo(s) ele admite e ter certeza de que o ISP aceita pelo menos um deles.

[arte: ver original p. 609]

[T]Tabela

Característica	**P0P3**	**IMAP**
Onde o protocolo é definido	RFC 1939	RFC 2060

Porta TCP usada	110	143
Onde o correio eletrônico é armazenado	PC do usuário	Servidor
Onde o correio eletrônico é lido	Off-line	On-line
Tempo de conexão exigido	Pequeno	Grande
Utilização de recursos do servidor	Mínima	Intensa
Várias caixas de correio	Não	Sim
Quem guarda cópias das caixas de correio	Usuário	ISP
Bom para usuários em trânsito	Não	Sim
Controle do usuário sobre o download	Pequeno	Grande
Downloads de mensagens parciais	Não	Sim
Quotas de disco constituem um problema	Não	Possível,
após algum tempo		
Implementação simples	Sim	Não
Suporte difundido	Sim	Crescendo

[F]Figura 7.17

[FL] Uma comparação entre POP3 e IMAP

[T4] Características de entrega

Independente de ser usado o POP3 ou o IMAP, muitos sistemas fornecem hooks (ganchos). Um recurso especialmente valioso para muitos usuários de correio eletrônico é a capacidade de estabelecer **filtros**. Os filtros são regras verificadas quando a mensagem chega ou quando o agente do usuário é inicializado. Cada regra especifica uma condição e uma ação. Por exemplo, uma regra poderia estabelecer que qualquer mensagem recebida do chefe deve ir para a caixa de correio número 1, que qualquer mensagem de um grupo selecionado de amigos deve ir para a caixa de correio número 2, e qualquer mensagem contendo certas palavras questionáveis na linha Subject (Assunto) deve ser descartada sem

comentários.

Alguns ISPs fornecem um filtro que divide automaticamente em categorias as mensagens de correio eletrônico recebidas, classificando-as como importantes ou como spam (lixo de correio eletrônico) e armazena cada mensagem na caixa de correio correspondente. Tais filtros em geral funcionam verificando primeiro se a origem é um spammer conhecido. Em seguida, eles costumam examinar a linha de assunto. Se centenas de usuários tiverem acabado de receber mensagens com a mesma linha de assunto, é provável que ela seja spam. Outras técnicas também são usadas para a detecção de spam.

Outro recurso de entrega oferecido com freqüência é a possibilidade de encaminhar (temporariamente) as mensagens recebidas para outro endereço. Esse endereço pode ser até mesmo um computador operado por um serviço comercial de paging (localização), que se comunica com o usuário por rádio ou satélite, exibindo a linha *Subject*: em seu pager.

Outro recurso comum de entrega final é a capacidade de instalar um **daemon de férias.** Trata-se de um programa que examina cada mensagem recebida e envia ao remetente uma resposta impessoal como:

[TD]Oi, estou de férias. Voltarei no dia 24 de agosto. Tenha um bom dia. [TN]

Essas respostas também podem especificar como tratar assuntos urgentes durante o período em questão, como entrar eme contato com outras pessoas para problemas específicos etc. A maior parte dos daemons de férias controla quem deverá receber as respostas prontas, impedindo o envio de uma segunda resposta para a mesma pessoa. Os daemons de melhor qualidade também verificam se as mensagens recebidas foram enviadas para uma lista de debate. Caso isso tenha ocorrido, eles não enviarão uma resposta pronta. (Provavelmente, as pessoas que enviam mensagens para grandes listas de debate durante o verão não querem receber centenas de respostas detalhando os planos de férias de

todo mundo.)

Recentemente, o autor deste livro utilizou uma forma radical de processamento de entrega quando enviou uma mensagem de correio eletrônico para uma pessoa que afirma receber 600 mensagens por dia. Sua identidade não será revelada aqui, pois metade dos leitores deste livro também poderá fazer o mesmo. Vamos chamá-la de John.

John instalou um robô de correio eletrônico que verifica todas as mensagens recebidas para descobrir se elas foram enviadas por um novo correspondente. Caso a resposta seja afirmativa, ele retorna uma mensagem pronta explicando que John não pode mais ler toda a correspondência pessoalmente e que, em vez disso, produziu um documento de FAQ (Frequently Asked Questions — perguntas freqüentes) pessoal que responde a muitas perguntas feitas a ele com freqüência. Em geral, os newsgroups têm FAQs, mas não as pessoas.

A FAQ de John informa seu endereço, os números de telefone e fax, e mostra como entrar em contato com a empresa em que ele trabalha. A FAQ explica como entrar em contato com ele para palestras e descreve onde obter artigos e documentos escritos por John. Na FAQ também há informações sobre produtos de software desenvolvidos por John, sobre uma conferência que ele está apresentando, um padrão criado por ele etc. Talvez essa estratégia seja necessária, mas é bem provável que uma FAQ pessoal também represente um símbolo de status.

[T4] Webmail

Um último tópico que vale a pena mencionar é o Webmail. Alguns sites da Web — por exemplo, Hotmail e Yahoo — fornecem serviço de correio eletrônico para qualquer pessoa que o deseje. Esses serviços funcionam da maneira descrita a seguir. Agentes de transferência de mensagens normais escutam a porta 25 em

busca de conexões de SMTP de entrada. Para entrar em contato, digamos, com o Hotmail, você tem de adquirir seu registro *MX* do DNS, por exemplo, digitando:

[TD] host -a -v hotmail.com [TN]

em um sistema UNIX. Suponha que o servidor de correio se denomine *mx10.hotmail.com*; então, digitando:

[TD] telnet mx10.hotmail.com 25 [TN]

você pode estabelecer uma conexão TCP sobre a qual os comandos SMTP podem ser enviados da maneira habitual. Até agora, nada de novo, a não ser o fato de que esses grandes servidores freqüentemente estão ocupados; então, talvez sejam necessárias várias tentativas para conseguir a aceitação de uma conexão TCP.

A parte interessante é a maneira como o correio eletrônico é entregue. Basicamente, quando o usuário vai até a página da Web de correio eletrônico, é apresentado um formulário em que o usuário deve fornecer um nome de login e uma senha. Quando o usuário clicar em Assinar (Sign In), o nome de login e a senha serão enviados ao servidor, que então efetuará a validação. Se o login for bem-sucedido, o servidor encontrará a caixa de correio do usuário e criará uma listagem semelhante à da Figura 7.8, formatada como uma página da Web em HTML. A página da Web é então enviada ao navegador para exibição. Muitos itens na página podem ser ativados por cliques; dessa maneira, as mensagens podem ser lidas, excluídas e assim por diante.

==

[T2] 7.3 A World Wide Web

A World Wide Web é uma estrutura arquitetônica que permite o acesso a documentos vinculados espalhados por milhões de máquinas na Internet. Em dez anos, ela deixou de ser um meio de distribuição de dados sobre física de alta energia para se tornar a aplicação que milhões de pessoas consideram ser "A Internet". Sua enorme popularidade se deve à sua interface gráfica colorida, de fácil utilização para principiantes. Além disso, ela oferece uma imensa variedade de informações sobre quase todos os assuntos imagináveis, desde aborígenes até zoologia.

A Web (também conhecida como **WWW**) teve início em 1989 no CERN, o centro europeu para pesquisa nuclear. O CERN tem vários aceleradores de partículas, nos quais grandes grupos de cientistas dos países europeus participantes desenvolvem pesquisas na área da física de partículas. Esses grupos são quase sempre compostos por membros de mais de meia dúzia de países diferentes. A maioria das experiências é altamente complexa e exige anos de planejamento para a construção dos equipamentos necessários. A Web nasceu da necessidade de fazer com que esses grupos de cientistas de diferentes nacionalidades pudessem colaborar uns com os outros através da troca de relatórios, plantas, desenhos, fotos e outros documentos.

A proposta inicial para uma teia de documentos vinculados veio de um físico do CERN, Tim Berners-Lee, em março de 1989. O primeiro protótipo (no modo texto) já era operacional um ano e meio depois. Em dezembro de 1991, foi realizada uma demonstração pública na conferência Hypertext ‘91, em San Antonio, no Texas.

Essa demonstração e a publicidade resultante atraíram a atenção de outros pesquisadores, o que levou Marc Andreessen, da University of Illinois, a iniciar o

desenvolvimento do primeiro navegador gráfico, o Mosaic, lançado em fevereiro de 1993. O Mosaic se tornou tão popular que, um ano mais tarde, Andreessen fundou sua própria empresa, a Netscape Communications Corp., cujo objetivo era desenvolver clientes, servidores e outros produtos de software para a Web. Quando a Netscape abriu seu capital ao público em 1995, os investidores, aparentemente pensando que ela seria a sucessora da Microsoft, pagaram 1,5 bilhão de dólares por suas ações. Esse recorde foi ainda mais surpreendente porque a empresa tinha apenas um produto, operava no vermelho e já havia anunciado que não previa a possibilidade de lucros para um futuro próximo. Nos três anos seguintes, o Netscape Navigator e o Internet Explorer da Microsoft se engajaram em uma "guerra de navegadores", cada um procurando incluir mais recursos (e, portanto, mais bugs) que o outro. Em 1998, o America Online comprou a Netscape Communications Corp. por 4,2 bilhões de dólares, encerrando assim a breve vida da Netscape como empresa independente.

Em 1994, o CERN e o MIT assinaram um acordo criando o World Wide Web Consortium (às vezes abreviado como **W3C**), uma organização voltada para o desenvolvimento da Web, a padronização de protocolos e para o incentivo à interoperabilidade entre os sites. Berners-Lee tornou-se o diretor do consórcio. Desde então, centenas de universidades e empresas juntaram-se ao consórcio. Embora existam agora mais livros sobre a Web do que se pode imaginar, o melhor lugar para obter informações atualizadas sobre a Web é (naturalmente) a própria Web. A home page do consórcio está no endereço *www.w3.org*. Os leitores interessados são levados por links a páginas que englobam todos os numerosos documentos e atividades de consórcio.

[T3] 7.3.1 Visão geral da arquitetura

Do ponto de vista dos usuários, a Web é uma vasta coleção mundial de

documentos, geralmente chamados **páginas da Web**, ou apenas **páginas.** Cada página pode conter links (vínculos) para outras páginas em qualquer lugar do mundo. Os usuários podem seguir um link (por exemplo, dando um clique sobre ele), que os levará até a página indicada. Esse processo pode ser repetido indefinidamente. A idéia de fazer uma página apontar para outra, agora chamada **hipertexto**, foi criada por um visionário professor de engenharia elétrica do MIT, Vannevar Bush, em 1945, bem antes da criação da Internet.

As páginas são visualizadas com o auxílio de um programa denominado **navegador**, dentre o Internet Explorer e o Netscape Navigator são dois exemplos conhecidos. O navegador busca a página solicitada, interpreta seu texto e seus comandos de formatação e exibe a página, formatada de modo apropriado, na tela do computador. Um exemplo de como isso é feito encontra-se na Figura 7.18(a). Assim como muitas outras páginas da Web, essa página começa com um título, contém algumas informações e termina com o endereço de correio eletrônico da pessoa que a mantém e atualiza. Os strings de texto que são links para outras páginas, chamados **hiperlinks**, freqüentemente estão destacados com sublinhados, por uma cor especial, ou ambos. Para seguir um link, o usuário posiciona o cursor sobre a área em destaque, o que faz o cursor mudar de forma, e clica sobre a área. Apesar de existirem navegadores sem recursos gráficos, como o Lynx, eles não são populares como os navegadores gráficos; portanto, vamos nos concentrar nesses últimos. Também já estão sendo desenvolvidos navegadores com comandos de voz.

Os usuários interessados no Department of Animal Psychology (Departamento de psicologia animal) podem aprender mais sobre ele dando um clique sobre seu nome (sublinhado). Em seguida, o navegador busca a página à qual o nome está vinculado e a apresenta na tela, como mostra a Figura 7.18(b). Também é possível clicar nos itens sublinhados para buscar outras páginas e assim por

diante. A nova página pode estar tanto na mesma máquina em que se encontra a primeira como em outra máquina em qualquer ponto do globo. O usuário não tem como saber. O navegador busca as páginas sem qualquer ajuda por parte do usuário. Se o usuário retornar à página principal, os links que já tiverem sido seguidos poderão estar destacados por uma linha pontilhada (e em uma outra cor), para distingui-los dos links que ainda não foram usados. Observe que, se você clicar sobre a linha *Campus Information* na página principal, nada acontecerá. Essa linha não está sublinhada, o que significa que não se trata de um link para outra página, e sim de um texto comum.

[arte: ver original p. 613]

[Dísticos]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 7.18

[FL] (a) Uma página da Web. (b) A página à qual se chega com um clique em Department of Animal Psychology

O modelo básico de funcionamento da Web é mostrado na Figura 7.19. Nessa figura, o navegador está exibindo uma página da Web na máquina cliente. Quando o usuário clica em uma linha de texto vinculada a uma página no servidor *abcd.com*, o navegador segue o hiperlink enviando uma mensagem ao servidor *abcd.com* na qual solicita a página. Ao chegar a página é exibida. Se essa página contiver um hiperlink para uma página no servidor *xyz.com* e ele for ativado por um clique, o navegador enviará uma solicitação da página a essa máquina, e assim por diante, indefinidamente.

[arte: ver original p. 614]

[Dísticos]

[1] Cliente

Página atual exibida pelo navegador

Hiperlink para abcd.com

Programa navegador

Conexão TCP

[2] Servidor abcd.com

Hiperlink para xyz.com

Disco Servidor da Web

A Internet

[3] Servidor xyz.com

Disco

Servidor da Web

[F]Figura 7.19

[FL] As partes do modelo da Web

[T4] O lado cliente

Agora vamos examinar o lado cliente da Figura 7.19 com mais detalhes. Em essência, um navegador é um programa que pode exibir uma página da Web e clicar com o mouse em itens na página exibida. Quando um item é selecionado, o navegador segue o hiperlink e busca a página selecionada. Por essa razão, o hiperlink incorporado precisa de um meio para nomear qualquer outra página na Web. As páginas são nomeadas com o uso de **URLs (Uniform Resource Locators).** Um URL típico é:

[TD] http://www.abcd.com/products.html [TN]

Explicaremos os URLs mais adiante neste capítulo. No momento, basta saber que um URL tem três partes: o nome do protocolo (*http*), o nome DNS da máquina em

que a página está localizada (*www.abcd.com*) e (normalmente) o nome do arquivo que contém a página (*products.html*).

Quando um usuário clica em um hiperlink, o navegador executa uma série de etapas em ordem para buscar a página indicada. Suponha que um usuário esteja navegando na Web e encontre um link sobre telefonia na Internet que aponta para a home page da ITU, *http://www.itu.org/home/index.html*. Vamos acompanhar as etapas que ocorrem quando esse link é selecionado.

1. O navegador determina o URL (verificando o que foi selecionado).
2. O navegador pergunta ao DNS qual é o endereço IP de *www.itu.org*.
3. O DNS responde com 156.106.192.32.
4. O navegador estabelece uma conexão TCP com a porta 80 em 156.106.192.32.
5. Em seguida, o navegador envia um comando solicitando o arquivo */home/index.html*.
6. O servidor *www.itu.org* envia o arquivo */home/index.html*.
7. A conexão TCP é encerrada.
8. O navegador exibe todo o texto de */home/index.html*.
9. O navegador busca e exibe todas as imagens que o arquivo contém.

Muitos navegadores exibem uma linha de status no rodapé da tela que indica qual etapa está sendo executada no momento. Dessa forma, quando o desempenho é fraco, o usuário pode verificar se a causa é falta de resposta do DNS, falta de resposta do servidor ou simplesmente congestionamento da rede durante a transmissão da página.

Para poder exibir a nova página (ou qualquer página), o navegador tem de reconhecer seu formato. Para permitir a todos os navegadores reconhecerem todas as páginas da Web, as páginas da Web são escritas em uma linguagem padronizada chamada HTML, que descreve páginas da Web. Vamos descrevê-la em detalhes mais adiante neste capítulo.

Embora um navegador seja basicamente um interpretador de HTML, a maioria dos navegadores tem numerosos botões e recursos para facilitar a navegação na Web. A maioria tem um botão para voltar à página anterior, um botão para ir até a página seguinte (que só opera depois que o usuário volta dessa página) e um botão para ir direto até a página inicial do próprio usuário. Muitos navegadores têm um botão ou um item de menu para definir um marcador (bookmark) em uma determinada página, e outro botão para exibir a lista de marcadores, tonando possível visitar outra vez qualquer página com apenas alguns cliques do mouse. As páginas também podem ser gravadas em disco ou impressas. Existem numerosas opções disponíveis para controlar o layout da tela e definir diversas preferências do usuário.

Além de terem texto comum (não sublinhado) e hipertexto (sublinhado), as páginas da Web também podem conter ícones, desenhos de linhas, mapas e fotografias. Cada um deles pode (opcionalmente) estar vinculado a outra página. Um clique em um desses elementos faz o navegador buscar a página vinculada e exibi-la na tela, como ocorre ao se clicar em um texto. No caso de imagens como fotos e mapas, a página a ser buscada em seguida pode depender da parte da imagem em que houve o clique.

Nem todas as páginas contêm HTML. Uma página pode consistir em um documento no formato PDF, um ícone em formato GIF, uma fotografia em formato JPEG, uma canção em formato MP3, um vídeo em formato MPEG ou qualquer um entre centenas de outros tipos de arquivos. Tendo em vista que as páginas HTML padrão podem se vincular a qualquer deles, o navegador tem um problema quando encontra uma página que não consegue interpretar.

Em vez de tornar os navegadores cada vez maiores construindo interpretadores para uma coleção de tipos de arquivos que cresce com rapidez, a maioria dos navegadores opta por uma solução mais geral. Quando um servidor retorna uma

página, ele também retorna algumas informações adicionais sobre a página. Essas informações incluem o tipo MIME da página (consulte a Figura 7.12). Páginas do tipo *text/html* são exibidas diretamente, como também as páginas criadas em alguns outros tipos internos. Se o tipo MIME não for um dos tipos internos, o navegador consulta sua tabela de tipos MIME para saber como exibir a página. Essa tabela associa um tipo MIME a um visualizador.

Há duas possibilidades: plug-ins e aplicações auxiliares. Um **plug-in** é um módulo de código que o navegador busca em um diretório especial no disco e instala como uma extensão do próprio navegador, como ilustra a Figura 7.20(a). Tendo em vista que os plug-ins funcionam dentro do navegador, eles têm acesso à página atual e podem modificar sua aparência. Depois que o plug-in termina seu trabalho (em geral, depois que o usuário passa para uma página da Web diferente), o plug-in é removido da memória do navegador.

[arte: ver original p. 616]

[Dísticos]

[1] Interface do navegador (usada pelo plug-in)

Interface do plug-in (usada pelo navegador)

[2] Máquina cliente

Navegador

Código básico O navegador funciona como um único processo

Plug-in

(a)

[3] Máquina cliente

Navegador Auxiliar

Processo 1 Processo 2

(b)

[F]Figura 7.20

[FL] (a) Um plug-in de navegador. (b) Uma aplicação auxiliar

Cada navegador tem um conjunto de procedimentos que todos os plug-ins devem implementar para que o navegador possa chamar o plug-in. Por exemplo, em geral existe um procedimento que o código básico do navegador chama para suprir o plug-in com dados para exibição. Esse conjunto de procedimentos é a interface do plug-in e é específico para o navegador.

Além disso, o navegador torna disponível um conjunto de seus próprios procedimentos para o plug-in, a fim de fornecer serviços aos plug-ins. Os procedimentos típicos na interface do navegador servem para alocar e liberar memória, exibir uma mensagem na linha de status do navegador e consultar o navegador sobre parâmetros.

Antes de utilizarmos um plug-in, ele deve ser instalado. O procedimento habitual de instalação é fazer o usuário ir até o Web site do plug-in e baixar um arquivo de instalação. No Windows, esse arquivo é normalmente um arquivo zip de auto-extração com a extensão *.exe*. Quando há um clique duplo no arquivo zip, é executado um pequeno programa associado ao arquivo zip. Esse programa descompacta o plug-in e o copia no diretório de plug-ins do navegador. Em seguida, ele efetua as chamadas apropriadas para registrar o tipo MIME do plug-in e para associar o plug-in a esse tipo. No UNIX, freqüentemente o instalador é um script do shell que manipula os processos de cópia e registro.

A outra maneira de estender um navegador é usar uma **aplicação auxiliar**, um programa completo que é executado como um processo separado. Essa aplicação é ilustrada na Figura 7.20(b). Tendo em vista que o auxiliar é um programa separado, ele não oferece nenhuma interface para o navegador e não faz uso de serviços do navegador. Em vez disso, em geral ele simplesmente aceita o nome de um arquivo de rascunho em que o arquivo de conteúdo é armazenado, abre o

arquivo e exibe o conteúdo. De modo geral, os auxiliares são grandes programas que existem de forma independente do navegador, como o Acrobat Reader da Adobe — para exibição de arquivos PDF — ou o Microsoft Word. Alguns programas (como o Acrobat) têm um plug-in que invoca o próprio auxiliar. Muitas aplicações auxiliares utilizam o tipo MIME *application*. Um número considerável de subtipos foi definido; por exemplo, *application/pdf* para arquivos PDF e *application/msword* para arquivos do Word. Desse modo, um URL pode apontar diretamente para um arquivo PDF ou do Word e, quando o usuário clicar sobre ele, o Acrobat ou o Word será inicializado automaticamente e receberá o nome de um arquivo de rascunho com o conteúdo a ser exibido. Como conseqüência, os navegadores podem ser configurados para lidar com um número virtualmente ilimitado de tipos de documentos, sem mudanças no navegador. Os modernos servidores da Web com freqüência são configurados com centenas de combinações de tipo/subtipo, e novas combinações são acrescentadas toda vez que um novo programa é instalado.

As aplicações auxiliares não se restringem a usar o tipo MIME *application*. Por exemplo, o Adobe Photoshop utiliza o tipo *image/x-photoshop*, e o RealOne Player é capaz de manipular *audio/mp3*.

No Windows, quando um programa é instalado no computador, ele registra os tipos MIME que deseja manipular. Esse mecanismo leva a conflitos quando vários visualizadores estão disponíveis para algum subtipo, como *video/mpg*. Isso ocorre porque o último programa a se registrar sobrescreve as associações (tipo MIME, aplicação auxiliar), capturando o tipo para ele mesmo. Em conseqüência disso, a instalação de um novo programa pode alterar a forma como um navegador lida com os tipos existentes.

No UNIX, em geral esse processo de inscrição não é automático. O usuário deve atualizar manualmente certos arquivos de configuração. Essa abordagem gera

mais trabalho, mas produz menos surpresas.

Os navegadores também podem abrir arquivos locais, em vez e buscá-los em servidores da Web remotos. Tendo em vista que os arquivos locais não têm tipos MIME, o navegador precisa de algum meio para determinar qual plug-in ou auxiliar deve usar para tipos diferentes de seus tipos internos, como *text/html* e *image/jpeg*. Para tratar arquivos locais, os auxiliares podem ser associados a uma extensão de arquivo, bem como a um tipo MIME. Com a configuração padrão, abrir *foo.pdf* significará abri-lo no Acrobat, e a ação de abrir *bar.doc* o abrirá no Word. Alguns navegadores utilizam o tipo MIME, a extensão de arquivo, e até mesmo informações obtidas do próprio arquivo para inferir o tipo MIME. Em particular, o Internet Explorer se baseia mais intensamente na extensão de arquivo que no tipo MIME, quando possível.

Nesse caso, também podem surgir conflitos, pois muitos programas estão dispostos — na verdade, ávidos — para tratar, digamos, arquivos *.mpg*. Durante a instalação, programas destinados a profissionais com freqüência exibem caixas de seleção com os tipos MIME e as extensões que eles estão preparados para manipular, a fim de permitir ao usuário selecionar as opções apropriadas e, assim, não sobrescrever associações existentes por engano. Os programas destinados ao mercado de consumo pressupõem que o usuário não tem uma indicação de qual seja o tipo MIME, e simplesmente captam tudo que podem sem levar em consideração o que foi feito pelos programas já instalados.

A habilidade de estender o navegador com um grande número de novos tipos é conveniente, mas também pode causar problemas. Quando o Internet Explorer busca um arquivo com a extensão *exe*, ele percebe que esse arquivo é um programa executável e, portanto, não tem auxiliar. A ação óbvia é executar o programa. Porém, isso poderia ser um enorme furo de segurança. Tudo que um site da Web malicioso precisa fazer é produzir uma página da Web com imagens

de, por exemplo, astros do cinema ou heróis do esporte, todos vinculados a um vírus. Um único clique em uma imagem provoca a busca de um programa executável desconhecido e potencialmente hostil, seguida por sua execução na máquina do usuário. Para evitar convidados indesejáveis como esse, o Internet Explorer pode ser configurado para ser seletivo ao executar programas desconhecidos de forma automática, mas nem todos os usuários entendem como administrar a configuração.

No UNIX, pode ocorrer um problema análogo com os scripts do shell, mas isso exige que o usuário instale de forma consciente o shell como um auxiliar. Felizmente, essa instalação é tão complicada que ninguém consegue realizá-la por acidente (e, na verdade, poucas pessoas podem realizá-la até mesmo de modo intencional).

[T4] O lado servidor

Já estudamos o lado cliente. Agora, vamos examinar o lado servidor. Como vimos antes, quando o usuário digita um URL ou clica em uma linha de hipertexto, o navegador analisa o URL e interpreta a parte entre *http://* e a barra seguinte como um nome DNS a ser pesquisado. Munido do endereço IP do servidor, o navegador estabelece uma conexão TCP para a porta 80 desse servidor. Em seguida, ele envia um comando contendo o restante do URL, que é o nome de um arquivo nesse servidor. O servidor então retorna o arquivo para ser exibido pelo navegador.

Em linhas gerais, um servidor da Web é semelhante ao servidor da Figura 6.6. Esse servidor, como um servidor da Web real, recebe o nome de um arquivo para pesquisar e retornar. Em ambos os casos, as etapas que o servidor executa em seu loop principal são:

1. Aceitar uma conexão TCP de um cliente (um navegador).

2. Obter o nome do arquivo solicitado.

3. Obter o arquivo (do disco).

4. Retornar o arquivo ao cliente.

5. Encerrar a conexão TCP.

Os servidores da Web modernos têm outras características mas, em essência, é isso que um servidor da Web faz.

Um problema com esse projeto é que toda solicitação exige a realização de um acesso ao disco para obter o arquivo. O resultado é que o servidor da Web não pode atender a mais solicitações por segundo do que o número de acessos de disco que ele pode realizar. Um disco SCSI de alta tecnologia tem um tempo de médio de acesso em torno de 5 ms, o que limita o servidor a no máximo 200 solicitações/segundo, ou até menos, caso grandes arquivos tenham de ser lidos com freqüência. No caso de um Web site importante, esse valor é muito baixo.

Uma melhoria óbvia (usada por todos os servidores da Web) é manter um cache na memória com os *n* arquivos mais recentemente usados. Antes de ir ao disco para obter um arquivo, o servidor verifica o cache. Se o arquivo estiver lá, ele poderá ser atendido diretamente da memória, eliminando assim o acesso ao disco. Embora o armazenamento efetivo no cache exija um grande volume de memória principal e algum tempo de processamento extra para verificar o cache e administrar seu conteúdo, a economia de tempo quase sempre compensa o overhead e as despesas.

A próxima etapa na elaboração de um servidor mais rápido é tornar o servidor multithreaded. Em um projeto, o servidor consiste em um módulo de front end que aceita todas as solicitações recebidas e *k* módulos de processamento, como mostra a Figura 7.21. Os $k + 1$ threads pertencem todos ao mesmo processo, de forma que todos os módulos de processamento têm acesso ao cache dentro do espaço de endereços do processo. Ao chegar uma solicitação, o front end a

recebe e cria um registro curto que a descreve. Em seguida, ele entrega o registro a um dos módulos de processamento. Em outro projeto possível, o front end é eliminado, e cada módulo de processamento tenta adquirir suas próprias solicitações; porém, nesse caso é necessário um protocolo de bloqueio para evitar conflitos.

[arte: ver original p. 619]

[Dísticos]

[1] Máquina do servidor da Web

Cache

[2] Módulo de processamento (thread)

[3] Front end

Solicitação recebida Resposta enviada

[F]Figura 7.21

[FL] Um servidor da Web multithreaded com um front end e módulos de processamento

O módulo de processamento verifica primeiro o cache para ver se o arquivo necessário está lá. Nesse caso, ele atualiza o registro com a finalidade de incluir um ponteiro para o arquivo no registro. Se ele não estiver no cache, o módulo de processamento inicia uma operação de disco para inserir o arquivo no cache (possivelmente descartando alguns outros arquivos armazenados no cache, a fim de abrir espaço para o novo arquivo). Quando o arquivo vem do disco, ele é inserido no cache e também é enviado de volta ao cliente.

A vantagem desse esquema é que, enquanto um ou mais módulos de processamento estão bloqueados esperando que uma operação de disco se complete (e, portanto, não consumindo tempo de CPU), outros módulos podem estar trabalhando ativamente em outras solicitações. É claro que, para obter

qualquer melhoria real em relação ao modelo de um único thread, é necessário ter vários discos, para que mais de um disco possa estar ocupado ao mesmo tempo. Com *k* módulos de processamento e *k* discos, o throughput pode ser no máximo *k* vezes maior que no caso de um servidor com um único thread e um disco.

Na teoria, um servidor com um único thread e *k* discos também poderia ter um ganho de *k* vezes, mas o código e a administração são muito mais complicados, pois as chamadas do sistema READ para bloqueio normal não podem ser usadas para acessar o disco. Com um servidor multithreaded, elas podem ser usadas porque, nesse caso, um READ só irá bloquear o thread que realizou a chamada, e não o processo inteiro.

Os modernos servidores da Web fazem mais que apenas aceitar nomes de arquivos e retornar arquivos. De fato, o processamento real de cada solicitação pode ficar bastante complicado. Por essa razão, em muitos servidores, cada módulo de processamento executa uma série de etapas. O front end repassa cada solicitação recebida ao primeiro módulo disponível, que então a executa usando algum subconjunto das etapas a seguir, dependendo de quais delas sejam necessárias para essa solicitação específica.

1. Resolver o nome da página da Web solicitada.
2. Autenticar o cliente.
3. Executar o controle de acesso no cliente.
4. Executar o controle de acesso na página da Web.
5. Verificar o cache.
6. Buscar a página solicitada no disco.
7. Determinar o tipo MIME para incluí-lo na resposta.
8. Cuidar de diversas tarefas.
9. Retornar a resposta ao cliente.

10. Criar uma entrada no log do servidor.

A etapa 1 é necessária, porque a solicitação recebida talvez não possa conter o nome real do arquivo como um string literal. Por exemplo, considere o URL *http://www.cs.vu.nl*, que tem um nome de arquivo vazio. Ele tem de ser expandido para formar algum nome de arquivo padrão. Além disso, os navegadores modernos podem especificar o idioma padrão do usuário (por exemplo, italiano ou inglês), o que torna possível para o servidor selecionar uma página da Web nesse idioma, se ela estiver disponível. Em geral, a expansão de nomes não é tão trivial quanto parece à primeira vista, devido a uma variedade de convenções sobre nomenclatura de arquivos.

A etapa 2 consiste em verificar a identidade do cliente. Essa etapa é necessária para páginas que não estão disponíveis para o público em geral. descreveremos um meio de fazê-lo mais adiante neste capítulo.

A etapa 3 verifica se há restrições sobre o fato da solicitação poder ou não ser atendida, considerando-se a identidade e a localização do cliente. A etapa 4 verifica se existem restrições de acesso associadas à página propriamente dita. Se um certo arquivo (por exemplo, *.htaccess*) estiver presente no diretório em que se localiza a página desejada, ele poderá restringir o acesso ao arquivo a domínios específicos; por exemplo, apenas a usuários de dentro da empresa.

As etapas 5 e 6 envolvem a obtenção da página. Na etapa 6 é necessário poder lidar com várias leituras de disco ao mesmo tempo.

A etapa 7 tem a finalidade de descobrir o tipo MIME a partir da extensão de arquivo, das primeiras palavras do arquivo, de um arquivo de configuração e talvez de outras fontes. A etapa 8 aborda um conjunto de tarefas diversas, como a elaboração de um perfil de usuário ou a obtenção de certas estatísticas.

A etapa 9 é aquele em que o resultado é enviado de volta, e a etapa 10 cria uma entrada no log do sistema para fins administrativos. Mais tarde, tais logs podem

ser pesquisados em busca de informações valiosas sobre o comportamento dos usuários; por exemplo, a ordem em que as pessoas acessam as páginas.

Se houver muitas solicitações a cada segundo, a CPU não será capaz de tratar a carga de processamento, independente de quantos discos sejam utilizados em paralelo. A solução é adicionar outros nós (computadores), possivelmente com discos replicados para que os discos se tornem o próximo gargalo. Isso nos leva ao modelo de **fazenda de servidores** da Figura 7.22. Um front end ainda aceita solicitações de entrada, mas as dispersa por várias CPUs, e não por vários threads, a fim de reduzir a carga sobre cada computador. As máquinas individuais podem elas próprias ter vários threads e pipelines como antes.

[arte: ver original p. 621]

[Dísticos]

[1] Roteador

[2] Nó de processamento (computador separado)

[3] Pipeline de thread

[4] LAN

[5] Front end

[F]Figura 7.22

[FL] Um grupo de servidores (server farm)

Um problema que ocorre com as fazendas de servidores é que não há mais um cache compartilhado, porque cada nó de processamento tem sua própria memória — a menos que seja usado um dispendioso multiprocessador de memória compartilhado. Um modo de levar em conta essa perda de desempenho é fazer um front end controlar o lugar para onde envia cada solicitação e enviar solicitações subseqüentes da mesma página para o mesmo nó. Isso faz de cada nó um especialista em certas páginas, pois o espaço de cache não é desperdiçado

pelo armazenamento de cada arquivo em cada cache.

Outro problema no caso das fazendas de servidores é que a conexão TCP do cliente termina no front end, e assim a resposta tem de passar pelo front end. Essa situação é representada na Figura 7.23(a), na qual a solicitação recebida (1) e a resposta enviada (4) passam ambas pelo front end. Às vezes, é utilizado um artifício denominado **handoff de TCP**, a fim de contornar esse problema. Com esse artifício, o ponto final do TCP é repassado ao nó de processamento, para que ele possa responder diretamente ao cliente, como mostra o item (3) na Figura 7.23(b). Esse handoff é feito de modo transparente para o cliente.

[arte: ver original p. 622]

[Dísticos]

[1] Nó de processamento

2 3

Front end

1 4

Do cliente Para o cliente

(a)

[2] Nó de processamento

2 3

Front end Para o cliente

1

Do cliente

(b)

[F]Figura 7.23

[FL] (a) Seqüência normal de mensagens de solicitação/resposta. (b) Seqüência quando é usado o handoff de TCP

[T4] URLs — Uniform Resource Locators

Dissemos repetidas vezes que as páginas da Web podem conter ponteiros para outras páginas da Web. Agora, chegou o momento de examinarmos com mais detalhes como esses ponteiros são implementados. Quando a Web foi criada, ficou aparente de imediato que fazer uma página apontar para outra página da Web exigia mecanismos de nomenclatura e localização de páginas. Em particular, três perguntas tinham de ser respondidas antes de uma página selecionada poder ser exibida:

1. Qual é o nome da página?
2. Onde a página está localizada?
3. Como a página pode ser acessada?

Se toda página recebesse de algum modo a atribuição de um nome exclusivo, não haveria nenhuma ambigüidade na identificação de páginas. Apesar disso, o problema não seria resolvido. Vamos considerar um paralelo entre pessoas e páginas. Nos Estados Unidos, quase todas as pessoas têm um número de registro de seguro social, que é um identificador único, pois não há duas pessoas com o mesmo número. Entretanto, munido apenas com o número do seguro social, é impossível descobrir o endereço da pessoa, e certamente não há como descobrir se devemos escrever para a pessoa em inglês, espanhol ou chinês. A Web apresenta basicamente os mesmos problemas.

A solução escolhida identifica as páginas de uma forma que resolve todos os três problemas ao mesmo tempo. A cada página é atribuído um **URL (Uniform Resource Locator)** que funciona como o nome universal da página. Os URLs têm três partes: o protocolo (também chamado esquema), o nome DNS da máquina em que a página está e um nome local que indica a página específica (normalmente, um nome de arquivo na máquina onde ele reside). Por exemplo, o Web site para o departamento do autor contém diversos vídeos sobre a

universidade e a cidade de Amsterdã. O URL para a página de vídeo é:

[TD] http://www.cs.vu.nl/video/index-en.html [TN]

Esse URL está dividido em três partes: o protocolo (*http*), o nome DNS do host (*www.cs.vu.nl*) e o nome do arquivo (*video/index-en.html*), com certos sinais de pontuação separando os fragmentos. O nome do arquivo é um caminho relativo ao diretório da Web padrão em *cs.vu.nl*.

Muitos sites têm atalhos internos para nomes de arquivos. Em vários sites, um nome de arquivo nulo representa como padrão a home page principal da organização. Normalmente, quando o arquivo nomeado é um diretório, isso implica um arquivo denominado *index.html*. Por fim, *~user/* poderia ser mapeado no diretório da WWW de *user*, e depois no arquivo *index.html* desse diretório. Portanto, a home page do autor pode ser acessada em:

[TD] http://www.cs.vu.nl/~ast/ [TN]

mesmo que o nome real do arquivo seja *index.html* em um certo diretório padrão.

Agora podemos ver como o hipertexto funciona. Para tornar um fragmento de texto que pode ser ativado por um clique ("clicável"), o criador da página deve fornecer dois itens de informação: o texto ativado por um clique a ser exibido e o URL da página de destino se o texto for selecionado. Explicaremos a sintaxe do comando mais adiante neste capítulo.

Quando o texto é selecionado, o navegador procura o nome do host usando o DNS. Agora, de posse do endereço IP do host, o navegador estabelece uma conexão TCP com o host. Através dessa conexão, ele envia o nome do arquivo, usando o protocolo especificado. Pronto! A página é recebida.

Esse esquema de URL é aberto, pois é simples para os navegadores o uso de vários protocolos para acessar diferentes tipos de recursos. Na verdade, foram definidos URLs para vários outros protocolos comuns. Formas ligeiramente

simplificadas dos mais comuns estão listadas na Figura 7.24.

Vamos examinar essa lista brevemente. O protocolo *http* é a linguagem natural da Web, que os servidores utilizam para se comunicar. **HTTP** significa **Hypertext Transfer Protocol.** Vamos examiná-lo em mais detalhes mais adiante neste capítulo.

[arte: ver original p. 624]

[T]Tabela

Nome	Usado para	Exemplo
http	Hipertexto (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Arquivo local	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	Artigo de newsgroup	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Enviar correio eletrônico	mailto:JohnUser@acm.org
telnet	Login remoto	telnet://www.w3.org:80

[F]Figura 7.24

[FL] Alguns URLs comuns

O protocolo *ftp* é usado para acessar arquivos por FTP, o protocolo de transferência de arquivos da Internet. O FTP já existe há mais de duas décadas e conquistou seu espaço. Um grande número de servidores FTP em todo o mundo permite que pessoas de qualquer lugar da Internet estabeleçam login e façam download de quaisquer arquivos armazenados no servidor FTP. A Web não muda esse fato; ela apenas torna a obtenção de arquivos por FTP mais fácil, pois o FTP tem uma interface um tanto misteriosa (embora seja mais eficiente que o HTTP; por exemplo, ele permite que um usuário na máquina *A* transfira um arquivo da

máquina *B* para a máquina *C*).

E possível acessar um arquivo local como uma página da Web, seja usando o protocolo *file* ou, mais simples ainda, usando apenas o nome do arquivo. Essa abordagem é semelhante ao uso do FTP, mas não exige um servidor. É óbvio que ela só funciona para arquivos locais, e não para arquivos remotos.

Bem antes de existir uma Internet, havia o sistema de notícias (news) da USENET. Ele consiste em cerca de 30.000 newsgroups (ou grupos de notícias) em que milhões de pessoas discutem uma ampla variedade de assuntos, publicando e lendo artigos relacionados ao tópico do newsgroup. O protocolo *news* pode ser usado para chamar um artigo de notícias como se ele fosse uma página da Web. Isso quer dizer que um navegador também é um newsreader. De fato, muitos navegadores têm botões ou itens de menus para tornar a leitura de artigos de notícias da USENET ainda mais fácil do que quando se empregam newsreaders padrão.

O protocolo *news* aceita dois formatos. O primeiro formato especifica um newsgroup e pode ser usado para obter uma lista dos artigos de um site de notícias pré-configurado. O segundo protocolo exige que seja fornecido o identificador de um artigo de newsgroup específico, nesse caso, *AA0134223112@cs.utah.edu*. O navegador busca então o artigo em seu site de notícias pré-configurado, usando o protocolo **NNTP (Network News Transfer Protocol)**. Não estudaremos o NNTP neste livro, mas ele se baseia livremente no SMTP e tem um estilo semelhante a este último.

O protocolo *gopher* era usado no sistema Gopher, criado na University of Minnesota e batizado com o nome das equipes de atletas daquela escola, os Golden Gophers (além de ser uma expressão de gíria que significa "go for", ou seja, vá buscar). O Gopher é muitos anos mais velho que a Web. Ele foi um mecanismo para recuperação de informações, conceitualmente semelhante à

própria Web, mas que só aceita textos e não imagens. Hoje ele é essencialmente obsoleto e quase não é mais usado.

Os dois últimos protocolos não servem realmente para buscar páginas na Web, mas são úteis mesmo assim. O protocolo *mailto* permite que os usuários enviem mensagens de correio eletrônico a partir de um navegador da Web. Para isso, basta dar um clique sobre o botão OPEN e especificar um URL, que consiste em *mailto:* seguido pelo endereço eletrônico do destinatário. A maioria dos navegadores responderá iniciando um programa de correio eletrônico com o endereço e alguns campos do cabeçalho já preenchidos.

O protocolo *telnet* é usado para estabelecer uma conexão on-line com uma máquina remota. Ele é utilizado da mesma maneira que o programa Telnet, o que não é surpresa, pois a maioria dos navegadores só chama o programa Telnet como uma aplicação auxiliar.

Em resumo, os URLs foram projetados não só para permitir que os usuários navegassem na Web, como também para que eles lidassem com FTP, notícias, Gopher, mensagens de correio eletrônico e telnet, tornando desnecessários os programas de interface do usuário especializados para esses outros serviços e integrando quase todos os acessos à Internet em um só programa, o navegador da Web. Se não fosse pelo fato desse esquema ter sido projetado por um pesquisador de física, ele poderia ser facilmente considerado o resultado de uma campanha do departamento de marketing de uma empresa de software.

Apesar de todas essas propriedades, o crescente uso da Web gerou um enfraquecimento inerente no esquema de URLs. Um URL aponta para um host específico. No caso de páginas intensamente referenciadas, é interessante ter várias cópias afastadas entre si, a fim de reduzir o tráfego da rede. O problema é que os URLS não oferecem um meio de fazer referência a uma página sem informar simultaneamente onde ela está. Não há como dizer: "Quero a página

xyz, mas não quero nem saber onde você a obterá". Para resolver esse problema e possibilitar a replicação de páginas, a IETF está trabalhando em um sistema de **URNs (Universal Resource Names).** Um URN pode ser entendido como um URL generalizado. Esse assunto ainda é o tema de muitas pesquisas atuais, embora exista uma proposta de sintaxe na RFC 2141.

[T4] Ausência de estados e cookies

Como vimos várias vezes, basicamente a Web não tem estados. Não existe nenhum conceito de sessão de login. O navegador envia uma solicitação a um servidor e recebe de volta um arquivo. Depois, o servidor esquece que já viu esse cliente específico.

No início, quando a Web era usada apenas para recuperar documentos publicamente disponíveis, esse modelo era bem adequado. Porém, à medida que a Web começou a adquirir outras funções, o modelo causou problemas. Por exemplo, alguns Web sites exigem que os clientes se registrem (e talvez paguem algum dinheiro) para usá-los. Isso faz surgir a questão de como os servidores podem distinguir entre solicitações de usuários registrados e todas as outras solicitações. Um segundo exemplo é o do e-commerce (comércio eletrônico). Se um usuário percorrer uma loja eletrônica colocando itens em seu carrinho de compras de vez em quando, como o servidor irá controlar o conteúdo do carrinho? Um terceiro exemplo é o dos portais da Web personalizados, como o Yahoo. Os usuários podem configurar uma página inicial detalhada, apenas com as informações que deseja (por exemplo, suas ações e seus clubes esportivos favoritos); porém, como o servidor consegue exibir a página correta se não sabe quem é o usuário?

À primeira vista, poderíamos pensar que os servidores conseguem localizar usuários observando seus endereços IP. No entanto, essa idéia não funciona. Em

primeiro lugar, muitos usuários trabalham em computadores compartilhados, especialmente nas empresas, e o endereço IP identifica apenas o computador, e não o usuário. Em segundo lugar, e até mesmo pior, muitos ISPs utilizam a NAT, e assim todos os pacotes de saída de todos os usuários têm o mesmo endereço IP. Do ponto de vista do servidor, todos os milhares de clientes dos ISPs utilizam o mesmo endereço IP.

Para resolver esse problema, a Netscape criou uma técnica muito criticada, denominada **cookies**. O nome deriva de uma antiga gíria dos programadores, na qual um programa chama um procedimento e recebe de volta algo que talvez precise apresentar mais tarde para conseguir a realização de algum trabalho. Nesse sentido, um descritor de arquivos do UNIX ou um descritor de objetos do Windows pode ser considerado um cookie. Os cookies foram formalizados mais tarde na RFC 2109.

Quando um cliente solicita uma página da Web, o servidor pode fornecer informações adicionais junto com a página solicitada. Essas informações podem incluir um cookie, um pequeno arquivo ou string (com 4 KB no máximo). Os navegadores armazenam os cookies oferecidos em um diretório de cookies no disco rígido do cliente, a menos que o usuário tenha desativado os cookies. Os cookies são arquivos ou strings, e não programas executáveis. Em princípio, um cookie poderia conter um vírus; no entanto, tendo em vista que os cookies são tratados como dados, oficialmente não existe nenhuma maneira do vírus funcionar e causar danos, embora sempre seja possível algum hacker explorar um bug do navegador para provocar a ativação de um cookie.

Um cookie pode conter até cinco campos, como mostra a Figura 7.25. O campo *Domain* informa de onde veio o cookie. Os navegadores devem confirmar que os servidores não estão mentindo a respeito de seu domínio. Cada domínio pode armazenar no máximo 20 cookies por cliente. O campo *Path* é um caminho na

estrutura de diretórios do servidor que identifica as partes da árvore de arquivos do servidor que podem usar o cookie. Freqüentemente, ele contém o símbolo / (barra), que representa a árvore inteira.

[arte: ver original p. 626]

[T]Tabela

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

[F]Figura 7.25

[FL] Alguns exemplos de cookies

O campo *Content* tem a forma *nome* = *valor*. Tanto *nome* quanto *valor* podem ser o que o servidor quiser. Esse campo é onde fica armazenado o conteúdo do cookie.

O campo *Expires* especifica quando o cookie irá expirar. Se esse campo estiver ausente, o navegador descartará o cookie ao terminar. Um cookie desse tipo é chamado **cookie não persistente.** Se forem fornecidas hora e data, o cookie é chamado **persistente** e será mantido até expirar. A hora de expiração (vencimento) é dada pelo horário de Greenwich (GMT). Para remover um cookie do disco rígido do cliente, basta o servidor enviá-lo novamente com uma data de expiração vencida.

Finalmente, o campo *Secure* pode ser ajustado para indicar que o navegador só deve retornar o cookie a um servidor seguro. Esse recurso é usado para e-commerce, transações bancárias e outras aplicações seguras.

Vimos como os cookies são obtidos; porém, como eles são usados? Antes e um navegador enviar uma solicitação de uma página a algum Web site, ele confere seu diretório de cookies para ver se existe algum cookie armazenado pelo domínio ao qual a solicitação é encaminhada. Em caso afirmativo, todos os cookies gerados por esse domínio são incluídos na mensagem de solicitação. Ao recebê-los, o servidor pode interpretá-los do modo que desejar.

Vamos examinar alguns usos possíveis para os cookies. Na Figura 7.25, o primeiro cookie foi definido por *toms-casino.com* e é usado para identificar o cliente. Quando o cliente se conectar na próxima semana para apostar novamente (e jogar fora mais algum dinheiro), o navegador enviará o cookie, para que o servidor saiba quem ele é. Munido da identificação do cliente, o servidor poderá pesquisar o registro do cliente em um banco de dados e usar essa informação para construir uma página da Web apropriada para exibição. Dependendo dos hábitos de apostas conhecidos do cliente, essa página pode consistir em uma rodada de pôquer, uma lista das corridas de cavalo de hoje ou uma máquina caça-níqueis.

O segundo cookie veio de *joes-store.com*. O cenário aqui é o do cliente que está vagando pela loja, procurando mercadorias. Ao encontrar uma pechincha, o cliente clica sobre o artigo e o servidor monta um cookie contendo o número de itens e o código dos produtos para enviar de volta ao cliente. À medida que o cliente continua a percorrer a loja, o cookie é transmitido a cada nova página solicitada. Conforme as compras se acumulam, o servidor adiciona as mercadorias ao cookie. Na figura, o carrinho contém três artigos; o último artigo é uma compra repetida. Finalmente, quando o cliente clica em *PROCEED TO CHECKOUT*, o cookie, que agora contém a lista completa de compras, é enviado junto com a solicitação. Desse modo, o servidor sabe exatamente o que foi adquirido.

O terceiro cookie é o cookie de um portal da Web. Quando o cliente clica em um link para o portal, o navegador transmite o cookie. Isso informa ao portal que ele deve elaborar uma página contendo os preços das ações da Sun Microsystems e da Oracle, e ainda os resultados das partidas de futebol do New York Jets. Tendo em vista que um cookie pode ter até 4 KB, há bastante espaço para preferências mais detalhadas relativas a manchetes de jornais, previsão do tempo local, ofertas especiais etc.

Os cookies também podem ser usados em benefício do próprio servidor. Por exemplo, suponha que um servidor queira saber quantos visitantes distintos ele recebeu e quantas páginas cada um percorreu antes de deixar o site. Quando a primeira solicitação entrar, ela não será acompanhada por nenhum cookie, e assim o servidor enviará de volta um cookie contendo *Counter* = 1. Cliques subseqüentes nesse site farão o cookie ser enviado de volta ao servidor. Toda vez que isso ocorre, o contador é incrementado e devolvido ao cliente. Controlando os contadores, o servidor pode verificar quantas pessoas desistiram depois de verem a primeira página, quantas observaram duas páginas e assim por diante.

Também há abusos relacionados aos cookies. Na teoria, os cookies só devem voltar ao site que os originou, mas os hackers têm explorado numerosos bugs nos navegadores para capturar cookies não destinados a eles. Tendo em vista que alguns sites de e-commerce inserem números de cartões de crédito em cookies, fica claro o potencial para abusos.

Um uso controvertido dos cookies tem a finalidade de reunir secretamente informações sobre os hábitos de navegação dos usuários. Ele funciona da maneira ilustrada a seguir. Uma agência de publicidade, digamos, a Sneaky Ads, entre em contato com Web sites importantes e insere banners anunciando os produtos de seus clientes corporativos nas páginas desses Web sites, pagando uma taxa aos proprietários do site. Em vez de fornecer ao site um arquivo GIF ou

JPEG para ser colocado em cada página, a agência lhes entrega um URL que será incluído na página. Cada URL entregue contém um número exclusivo na parte de arquivo, como:

[TD] http://www.sneaky.com/382674902342.gif [TN]

Quando um usuário visita pela primeira vez uma página *P* que contém esse anúncio, o navegador busca o arquivo HTML. Em seguida, o navegador examina o arquivo HTML e encontra o link para o arquivo de imagem em *www.sneaky.com* e, em seguida, solicita a imagem. Um arquivo GIF contendo um anúncio é retornado, juntamente com um cookie contendo uma ID de usuário exclusiva, 3627239101, na Figura 7.25. A Sneaky registra o fato de que o usuário com essa ID visitou a página *P*. É fácil fazer isso, pois o arquivo solicitado (*382674902342.gif*) só é referenciado na página *P*. É claro que o anúncio real pode aparecer em milhares de páginas, mas cada vez com um nome de arquivo diferente. Provavelmente a Sneaky ganha alguns centavos do fabricante do produto toda vez que transmite o anúncio.

Mais tarde, quando o usuário visitar outra página da Web contendo qualquer anúncio da Sneaky, após o navegador buscar o arquivo HTML no servidor, ele encontrará o link para, digamos, *http://www.sneaky.com/493654919923.gif* e solicitará esse arquivo. Como ele já tem um cookie do domínio *sneaky.com*, o navegador incluirá o cookie da Sneaky contendo a ID do usuário. Agora a Sneaky sabe qual foi a segunda página que o usuário visitou.

Depois de algum tempo, a Sneaky poderá elaborar um perfil completo dos hábitos de navegação do usuário, ainda que ele jamais tenha clicado em qualquer dos anúncios. É claro que a agência ainda não tem o nome do usuário (embora tenha seu endereço IP, o que pode ser suficiente para deduzir o nome a partir de outros bancos de dados). Porém, se o usuário fornecer seu nome a algum site que coopere com a Sneaky, haverá um perfil completo juntamente com um nome

disponível para ser vendido a quem quiser comprá-lo. A venda dessas informações pode ser lucrativa o suficiente para a Sneaky colocar mais anúncios em mais Web sites e assim coletar mais informações. A parte mais insidiosa de tudo isso é que a maioria dos usuários não tem o menor conhecimento dessa coleta de informações e pode até pensar que está segura, porque nunca clicou em qualquer dos anúncios.

E se a Sneaky quiser ser superfurtiva, o anúncio não precisa ser um banner clássico. Um "anúncio" consistindo em um único pixel na cor de fundo (e, desse modo, invisível) tem exatamente o mesmo efeito de um banner: ele exige que o navegador vá buscar a imagem GIF de 1 × 1 pixel e envie todos os cookies originários do domínio do pixel.

Para manter uma certa sensação de privacidade, alguns usuários configuram seus navegadores para rejeitar todos os cookies. Porém, isso pode trazer problemas com Web sites legítimos que utilizam cookies. Para resolver essa questão, às vezes os usuários instalam software para eliminar cookies. Esses produtos de software são programas especiais que inspecionam cada cookie recebido assim que ele chega e aceitam ou descartam o cookie, dependendo das opções fornecidas pelo usuário (por exemplo, sobre os Web sites confiáveis). Isso dá ao usuário um controle preciso sobre os cookies que devem ser aceitos e rejeitados. Os navegadores modernos, como Mozilla (*www.mozilla.org*), têm controles internos elaborados sobre cookies, definidos pelo usuário.

[T3] 7.3.2 Documentos estáticos da Web

A base da Web é a transferência de páginas da Web do servidor para o cliente. Em sua forma mais simples, as páginas da Web são estáticas, isto é, são apenas arquivos que ficam armazenados em algum servidor esperando o momento de serem recuperados. Nesse contexto, até mesmo um vídeo é uma página da Web

estática, porque é simplesmente um arquivo. Nesta seção, examinaremos páginas da Web estáticas em detalhes. Na próxima, examinaremos o conteúdo dinâmico.

[T4] HTML — HyperText Markup Language

Atualmente, as páginas da Web são escritas em uma linguagem denominada **HTML (HyperText Markup Language).** A HTML permite que os usuários produzam páginas da Web que incluem texto, gráficos e ponteiros para outras páginas da Web. A HTML é uma linguagem de marcação, ou seja, uma linguagem para descrever como os documentos devem ser formatados. O termo "marcação" vem da época em que os editores realmente marcavam os documentos para informar ao impressor — naquele tempo, uma pessoa — que fontes usar e assim por diante. Portanto, as linguagens de marcação contêm comandos explícitos de formatação. Por exemplo, em HTML, **significa início do modo negrito, e** significa fim do modo negrito. A vantagem de uma linguagem de marcação sobre outra sem marcação explícita é a maior facilidade para criar um navegador destinado à linguagem de marcação: o navegador só precisa entender os comandos de marcação. TeX e troff são outros exemplos muito conhecidos de linguagens de marcação.

Embutindo todos os comandos de marcação em cada arquivo de HTML e padronizando-os, torna-se possível para qualquer navegador da Web ler e reformatar qualquer página da Web. A capacidade de reformatar páginas da Web depois de recebê-las é crucial, porque uma página pode ter sido produzida em uma janela de 1600 × 1200 com cores de 24 bits, mas talvez tenha de ser exibida em uma janela de 640 × 320 configurada para cores de 8 bits.

Faremos uma breve introdução à HTML, apenas para lhe dar uma idéia de sua aparência. Embora sem dúvida seja possível escrever documentos HTML com qualquer editor padrão, e muitas pessoas o fazem, também é possível usar

editores ou processadores de textos especiais de HTML que fazem a maior parte do trabalho (mas que, ao mesmo tempo, oferecem ao usuário menos controle sobre todos os detalhes do resultado final).

Uma página da Web consiste em um cabeçalho e um corpo entre as **tags** (comandos de formatação) [TD]<html>[TN] e [TD]</html>[TN], embora a maioria dos navegadores não reclame se essas tags não estiverem presentes. Como podemos ver na Figura 7.26(a), o cabeçalho começa e termina com as tags [TD]<head>[TN] e [TD]</head>[TN] respectivamente, enquanto com o corpo é delimitado pelas tags [TD]<body>[TN] e [TD]</body>[TN]. Os strings entre as tags são chamados **diretivas**. A maioria das tags de HTML tem esse formato, ou seja, [TD]<algo>[TN] para marcar o início de alguma coisa e [TD]</algo>[TN] para marcar seu fim. A maioria dos navegadores tem um item de menu VIEW SOURCE ou algo parecido. Ao selecionar esse item, o código-fonte de HTML da página é exibido em lugar de sua versão formatada.

As tags não fazem distinção entre maiúsculas/minúsculas. Portanto, [TD]<head>[TN] e [TD]<HEAD>[TN] têm o mesmo significado, embora versões mais recentes do padrão exijam apenas letras minúsculas. O verdadeiro layout do documento HTML é irrelevante. Os analisadores de HTML ignoram espaços em branco e retornos de cursor, pois têm de reformatar o texto de modo a encaixá-lo na área de exibição atual. Conseqüentemente, o espaço em branco pode ser acrescentado à vontade para tornar os documentos HTML mais legíveis, algo de que a maioria dos documentos precisa muito. Outra conseqüência é que as linhas em branco não podem ser usadas para separar parágrafos, pois elas são simplesmente ignoradas. É necessário usar uma tag explícita.

Algumas tags têm parâmetros (com nomes, chamados **atributos**). Por exemplo:

[TD]<img src="abc" alt="foobar">[TN]

é uma tag [TD]<img>[TN] com o parâmetro *src* definido com valor igual a *abc* e

com o parâmetro *alt* definido com valor igual a *foobar*. Para cada tag, o padrão HTML fornece uma lista de parâmetros possíveis, se houver, e seu significado. Como cada parâmetro tem um nome, a ordem em que eles são apresentados não é significativa.

Em termos técnicos, os documentos HTML são escritos no conjunto de caracteres ISO 8859-1 Latin-1 mas, para os usuários cujos teclados só aceitam a tabela ASCII, há seqüência de escape para caracteres especiais, como é o caso de à. A lista de caracteres especiais é fornecida no padrão. Todos começam com o sinal & e terminam com um ponto-e-vírgula. Por exemplo, [TD] [TN] produz um espaço, [TD]à[TN] produz à e [TD]é[TN] produz é. Como os sinais <, > e & têm significados especiais, eles só podem ser expressos com suas seqüências de escape, [TD]<[TN] [TD]>[TN] e [TD]&[TN], respectivamente.

O item mais importante do cabeçalho é o título, delimitado por [TD]<title>[TN] e [TD]</title>[TN], mas alguns tipos de metainformações também podem estar presentes. O título propriamente dito não aparece na página. Alguns navegadores o utilizam para identificar a janela da página.

Vamos examinar agora outras características ilustradas na Figura 7.26. Todas as tags usadas na Figura 7.26 e algumas outras são mostradas na Figura 7.27. Os cabeçalhos são gerados por uma tag [TD]<h*n*>[TN], onde *n* é um dígito no intervalo 1 a 6. Desse modo, [TD]<h1>[TN] é o cabeçalho mais importante; [TD]<h6>[TN] é o menos importante. Cabe ao navegador representar essa hierarquia de maneira apropriada na tela. De modo geral, os cabeçalhos de números mais baixos serão exibidos com fontes maiores e mais grossas. O navegador também pode optar por usar cores diferentes para cada nível de cabeçalho. Em geral, os cabeçalhos [TD]<h1>[TN] são escritos com fontes grandes, em negrito e com pelo menos uma linha em branco acima e outra

abaixo. Em contraste, os cabeçalhos [TD]<h2>[TN] são mostrados com uma fonte menor e com menos espaço acima e abaixo.

[arte: ver original da p. 631]

[TD]

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a>
   <li> <a href="http://widget.com/products/little"> Little widgets </a>
</ul>
<h2> Telephone numbers </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax: 1-415-765-4321
</ul>
</body>
</html> [TN]
```

(a)

Welcome to AWI’s Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page.

We hope *you* will find all the information you need here.

Below we have links to intormation about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.

Product Information Big widgets Little widgets

Telephone numbers 1-800-WIDGETS 1-415-765-4321

(b)

[F]Figura 7.26

[FL] (a) O código HTML correspondente a um exemplo de página da Web. (b) A página formatada

As tags [TD]<b>[TN] e [TD]<i>[TN] são usadas para dar início aos modos negrito e itálico, respectivamente. Se o navegador não for capaz de apresentar caracteres em negrito e em itálico, ele deverá usar outro método para representá-los; por exemplo, usando uma cor diferente para cada um ou talvez vídeo reverso.

A HTML oferece vários mecanismos para criação de listas, incluindo listas aninhadas. As listas são iniciadas com [TD]<ul>[TN] ou [TD]<ol>[TN], sendo [TD]<li>[TN] usada para marcar o início dos itens em ambos os casos. A tag [TD]<ul>[TN] dá início a uma lista não ordenada. Os itens da lista, marcados com a tag [TD]<li>[TN] no código-fonte, aparecem com bullets ([B]) à esquerda deles. A variante desse mecanismo é [TD]<ol>[TN] que se destina a listas ordenadas. Quando essa tag é usada, os itens [TD]<li>[TN] são numerados pelo navegador. Embora utilizem tags iniciais e finais diferentes, [TD]<ul>[TN] e [TD]<ol>[TN] têm a mesma sintaxe e apresentam resultados semelhantes.

As tags [TD]
[TN], [TD]<p>[TN] e [TD]<hr>[TN] indicam uma delimitação entre seções do texto. O formato preciso pode ser determinado pela folha de estilos (veja a seguir) associada à página. A tag [TD]
[TN] apenas força uma

quebra de linha. De modo geral, os navegadores não inserem uma linha em branco depois de [TD]
[TN]. Ao contrário, [TD]<p>[TN] inicia um parágrafo, o que deve, por exemplo, inserir uma linha em branco e possivelmente algum tipo de recuo. (Na teoria, a tag [TD]</p>[TN] existe para marcar o fim de um parágrafo, mas raramente é usada; a maioria dos autores de HTML sequer sabe que ela existe.) Por último, [TD]<hr>[TN] força uma quebra de linha e desenha uma linha horizontal na tela.

A HTML permite a inclusão de imagens em linha em uma página da Web. A tag [TD]<img>[TN] especifica que uma imagem será carregada nessa posição na página. Ela pode ter vários parâmetros. O parâmetro *src* mostra o URL da imagem. O padrão HTML não especifica que formatos gráficos são permitidos. Na prática, todos os navegadores aceitam arquivos GIF e JPEG. Os navegadores têm liberdade para aceitar outros formatos, mas essa extensão é uma faca de dois gumes. Se estiver acostumado com um navegador que aceita, digamos, arquivos BMP, o usuário poderá incluir esses arquivos em suas páginas da Web e depois ficar surpreso ao descobrir que outros navegadores ignoram sua maravilhosa arte.

Outros parâmetros de [TD]<img>[TN] são *align*, que controla o alinhamento da imagem em relação à linha de base do texto (*top*, *middle*, *bottom*), *alt*, que oferece um texto a ser usado em lugar da imagem quando o usuário tiver desativado a exibição de imagens e *ismap*, um flag que indica que a imagem é um mapa ativo (ou seja, uma figura que pode ser acionada por um clique).

Por último, chegamos aos hiperlinks, que utilizam as tags [TD]<a>[TN] (âncora) e [TD][TN]. Da mesma forma que [TD]<img>[TN], [TD]<a>[TN] tem vários parâmetros, incluindo *href* (o URL) e *name* (o nome do hiperlink). O texto entre [TD]<a>[TN] e [TD][TN] é apresentado na tela. Se for selecionado, o hiperlink levará a uma nova página. Também é permitido incluir uma imagem

[TD]<img>[TN]; nesse caso, um clique sobre a imagem também ativa o hiperlink.

Como exemplo, vamos considerar o seguinte fragmento de HTML:

[TD] NASA's home page [TN]

Quando a página correspondente a esse fragmento for exibida, o que aparecerá na tela será:

[TD]NASA's home page [TN]

[arte: ver original p. 633]

[T]Tabela

Tag	**Descrição**
[TD]<html> ... </html>[TN]	Declara a página da Web a ser criada em HTML
[TD]<head> ... </head>[TN]	Delimita o cabeçalho da página
[TD]<title> ... </title>[TN]	Define o título (não exibido na página)
[TD]<body> </body>[TN]	Delimita o corpo da página
[TD]<h*n*> ... </h*n*>[TN]	Delimita um cabeçalho de nível *n*
[TD] ... [TN]	Define ... em negrito
[TD]<i> ... </i>[TN]	Define ... em itálico
[TD]<center> ... </center>[TN]	Centraliza ... na página horizontalmente
[TD] [TN]	Delimita uma lista não ordenada (com bullets)
[TD] ... [TN]	Delimita uma lista numerada
[TD] ... [TN]	Dá início a um item em uma lista ordenada ou numerada
[TD] [TN]	Força uma quebra de linha
[TD]<p>[TN]	Inicia um parágrafo
[TD]<hr>[TN]	Insere uma linha (régua) horizontal
[TD]<img src="...">[TN]	Exibe uma imagem
[TD] ... [TN]	Define um hiperlink

[F]Figura 7.27

[FL] Uma seleção de tags comuns de HTML. Algumas podem ter outros parâmetros

Se depois disso o usuário clicar sobre esse texto, o navegador buscará imediatamente a página cujo URL é *http://www.nasa.gov* e a exibirá.

Considere este segundo exemplo:

[TD]<a href"http://www.nasa.gov"> <img src="shuttle.gif" alt="NASA">[TN]

Ao ser apresentada, essa página mostrará uma foto (por exemplo, do ônibus espacial). Ao clicar sobre a foto, você irá para a home page da NASA, como acontecerá ao clicar no texto sublinhado do exemplo anterior. Se o usuário tiver desativado a exibição automática de imagens, o texto NASA será exibido no lugar da foto.

A tag [TD]<a>[TN] pode usar um parâmetro *name* para incluir um hiperlink. Dessa forma, será possível fazer referência a ela dentro da página. Por exemplo, algumas páginas começam com um sumário cujos itens podem ser acionados por cliques do mouse. Ao clicar sobre um dos itens do sumário, o usuário saltará para a seção correspondente da página.

A HTML continua a evoluir. A HTML 1.0 e a HTML 2.0 não tinham tabelas, mas elas foram incluídas na HTML 3.0. Uma tabela de HTML consiste em uma ou mais linhas, cada uma formada por uma ou mais **células**. As células podem conter uma ampla variedade de itens, incluindo texto, figuras, ícones, fotografias e mesmo outras tabelas. As células podem ser mescladas para formar, por exemplo, um título que se estende por várias colunas. Os autores de páginas têm um controle limitado sobre o layout, incluindo o alinhamento, estilos de bordas e margens de células, mas os navegadores definem a apresentação final das tabelas.

A definição de uma tabela HTML é ilustrada na Figura 7.28(a), e uma

representação possível é mostrada na Figura 7.28(b). Esse exemplo simplesmente mostra algumas características básicas das tabelas HTML. As tabelas são iniciadas pela tag [TD]<table>[TN]. Outras informações podem ser fornecidas para descrever propriedades gerais da tabela.

A tag [TD]<caption>[TN] pode ser utilizada com o objetivo de oferecer uma legenda para uma figura. Cada linha começa com uma tag [TD]<tr>[TN]. As células individuais são marcadas por [TD]<th>[TN] ou [TD]<td>[TN]. A distinção é feita para permitir ao navegador usar diferentes representações para elas, como fizemos no exemplo.

Numerosos atributos também são permitidas nas tabelas. Eles incluem meios para especificar o alinhamento vertical e horizontal de células, a justificação dentro de uma célula, bordas, agrupamentos de células, unidades e muito mais.

Na HTML 4.0, foram acrescentados novos recursos. Entre eles, encontram-se recursos de acessibilidade para usuários com necessidades especiais, incorporação de objetos (uma generalização da tag [TD]<img>[TN] para que outros objetos também possam ser incorporados em páginas), suporte para linguagens de scripts (para permitir conteúdo dinâmico) e vários outros.

Quando um Web site é complexo e consiste em muitas páginas produzidas por vários autores que trabalham na mesma empresa, freqüentemente é interessante ter um meio de impedir que diferentes páginas tenham uma aparência distinta. Esse problema pode ser resolvido usando-se **folhas de estilos**. Quando as folhas de estilos são utilizadas, as páginas individuais deixam de usar estilos físicos, como negrito e itálico. Em vez disso, os autores de páginas utilizam estilos lógicos como [TD]<dn>[TN] (definição), [TD]<em>[TN] (ênfase fraca), [TD]<strong>[TN] (ênfase forte) e [TD]<var>[TN] (variáveis de programa). Os estilos lógicos são definidos na folha de estilos, referida no início de cada página. Desse modo, todas as páginas têm o mesmo estilo e, se o webmaster decidir

mudar [TD]<strong>[TN] de itálico com 14 pontos em azul para negrito com 18 pontos na cor rosa choque, bastará alterar uma definição para converter o Web site inteiro. Uma folha de estilos pode ser comparada a um arquivo [TD]#include[TN] em um programa C: a mudança de uma única definição de macro provoca a alteração em todos os arquivos de programa que incluem o cabeçalho.

[T4] Formulários

A HTML 1.0 só permitia a comunicação em um sentido. Os usuários podiam acessar páginas dos provedores de informações, mas era difícil enviar informações no sentido inverso. À medida que mais e mais organizações comerciais começaram a usar a Web, a necessidade de um tráfego em duas vias tornou-se bem maior. Por exemplo, muitas empresas desejavam ter a possibilidade de receber encomendas de seus produtos através de suas páginas da Web, os fornecedores de software queriam distribuir seus produtos pela Web de forma que os clientes pudessem preencher seus cartões de registro eletronicamente, e as empresas que ofereciam serviços de pesquisa na Web queriam que seus clientes pudessem digitar palavras-chaves para as pesquisas. Essas necessidades levaram à inclusão de **formulários** a partir da HTML 2.0. Os formulários têm caixas ou botões que permitem aos usuários fornecer informações ou fazer escolhas e retornar esses dados para o proprietário da página. A tag [TD]<input>[TN] é usada com essa finalidade. Ela tem uma variedade de parâmetros para determinar o tamanho, a natureza e o uso da caixa exibida. Os formulários mais comuns são campos em branco para a digitação de dados, caixas que podem ser selecionadas, mapas ativos e botões *submit*. O exemplo da Figura 7.29 ilustra algumas dessas opções.

[arte: ver original da p. 635]

[TD]

```
<html>

<head> <title> Um exemplo  de página com uma tabela </title> </head>

<body>

<table border=1 rules=all>

<caption> Algumas diferenças entre versões de HTML </caption>

<col align=left>

<col align=center>

<col align=center>

<col align=center>

<col align=center>

<tr> <th>Item  <th>HTML 1.0  <th>HTML 2.0  <th>HTML 3.0 <th>HTML 4.0
</tr>

<tr> <th> Hiperlinks <td> x <td> x <td> x <td> x </tr>

<tr> <th> Imagens <td> x <td> x <td> x <td> x </tr>

<tr> <th> Listas <td> x <td> x <td> x <td> x </tr>

<tr> <th> Imagens e mapas ativos <td>   <td> x <td> x <td> x </tr>

<tr> <th> Formulários <td>   <td> x <td> x <td> x </tr>

<tr> <th> Equações <td>   <td>   <td> x <td> x </tr>

<tr> <th> Barras de ferramentas <td>   <td>   <td> x <td> x
</tr>

<tr> <th> Tabelas <td>   <td>   <td> x <td> x </tr>

<tr> <th> Recursos de acessibilidade <td>   <td>   <td>  
<td> x </tr>

<tr> <th> Incorporação  de objetos <td>   <td>   <td>  
<td> x </tr>

<tr> <th> Criação de scripts <td>   <td>   <td>   <td> x
```

</tr>

</table>

</body>

</html> [TN]

(a)

Algumas diferenças entre versões de HTML

Item	**HTML 1.0**	**HTML 2.0**	**HTML 3.0**	**HTML 4.0**
Hiperlinks	X	X	X	X
Imagens	X	X	X	X
Listas	X	X	X	X
Imagens e mapas ativos		X	X	X
Formulários		X	X	X
Equações			X	X
Barras de ferramentas			X	X
Tabelas			X	X
Recursos de acessibilidade				X
Incorporação de objetos				X
Criação de scripts				X

[F]Figura 7.28

[FL] (a) Uma tabela HTML. (b) Uma possível interpretação dessa tabela

Vamos iniciar nossa descrição de formulários examinando esse exemplo. Como todos os formulários, esse é delimitado pelas tags [TD]<form>[TN] e [TD]</form>[TN]. Os textos não delimitados por uma tag são exibidos na tela sem qualquer modificação. Todas as tags comuns (como, por exemplo, [TD]<b>[TN]) são permitidas nos formulários. Três tipos de caixas de entrada são usadas nesse formulário.

O primeiro tipo de caixa de entrada é a que segue o texto "Name". Esse campo pode conter até 46 caracteres e espera que o usuário digite um string, que será armazenado na variável *customer* para processamento posterior. A tag [TD]<p>[TN] instrui o navegador a exibir textos e caixas subseqüentes na próxima linha, mesmo que haja espaço na linha atual. Ao usar [TD]<p>[TN] e outras tags de layout, o autor da página pode controlar a aparência do formulário na tela.

A próxima linha do formulário solicita o endereço do usuário, tem 40 caracteres de largura e também armazena somente essa informação. Em seguida, vem uma linha que solicita os nomes da cidade, do estado e do país. A tag [TD]<p>[TN] não é usada entre esses campos; portanto, o navegador exibirá todos eles na mesma linha, se houver espaço. Do ponto de vista do navegador, esse parágrafo contém apenas seis itens: três strings que se alternam com três caixas. Ele os exibe de forma linear, da esquerda para a direita, passando para a outra linha sempre que a linha atual não puder conter o item seguinte. Assim, é fácil prever que em uma tela de 1600 × 1200, os três strings e suas caixas correspondentes aparecerão na mesma linha, mas em uma tela de 1024 × 768 talvez eles estejam divididos em duas linhas. No pior cenário possível, a palavra "País" estará no fim de uma linha e sua caixa no início da linha seguinte.

A próxima linha pede o número do cartão de crédito e sua data de validade. Transmitir o número do cartão de crédito pela Internet é um procedimento que só deve ser realizado se forem tomadas todas as medidas de segurança cabíveis. Discutiremos algumas delas no Capítulo 8.

Depois da data de validade, encontramos um novo recurso: os botões de rádio. Eles são utilizados quando é necessário optar entre duas ou mais alternativas. O modelo imaginário aqui é um rádio de carro com meia dúzia de botões para sintonizar as estações. O navegador exibe essas caixas em um formulário que

permite ao usuário selecionar e cancelar as opções com um clique sobre elas (ou usando o teclado). Ao clicar sobre uma delas, o usuário desativa todas as outras do mesmo grupo. A apresentação visual cabe ao navegador. O campo Widget size também utiliza dois botões de rádio. Os dois grupos se distinguem por seu campo *name*, e não por um escopo estático que utilize algo como [TD]<radiobutton> ... </radiobutton>[TN].

Os parâmetros *value* são usados para indicar que botão de rádio foi pressionado. Dependendo de qual das opções de cartão de crédito o usuário selecionou, a variável *cc* será definida como o string "mastercard" ou "visacard".

Depois dos dois conjuntos de botões de rádio, chegamos à opção de transporte, representada por uma caixa de seleção do tipo *checkbox*. Ela pode estar selecionada ou não. Ao contrário dos botões de rádio, onde apenas uma opção deve ser escolhida, cada caixa de seleção do tipo *checkbox* pode ser ou não selecionada, de forma independente de todas as outras. Por exemplo, ao encomendar uma pizza na página da Web da Electropizza, o usuário pode escolher sardinhas *e* cebolas *e* abacaxi (se ele suportar essa mistura), mas não pode escolher pequena *e* média *e* grande para a mesma pizza. As coberturas das pizzas devem estar representadas por caixas do tipo *checkbox*, enquanto o tamanho da pizza estaria em um conjunto de botões de rádio.

[arte: ver original da p. 637]

[TD]

```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
```

```
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html> [TN]
```

(a)

Atenção, produção!

Favor reproduzir integralmente a figura do O.A.

(b)

[F]Figura 7.29

[FL] (a) A versão HTML para um formulário de pedido. (b) A página formatada

Cabe observar que para listas muito longas, a partir das quais deve-se escolher um item, os botões de rádio são um tanto inconvenientes. Assim, são oferecidas as tags [TD]<select>[TN] e [TD]</select>[TN], entre as quais pode-se incluir uma lista de alternativas, seguindo o padrão dos botões de rádio (a menos que lhe seja dado o parâmetro *multiple*; nesse caso, a lista segue o padrão das caixas de

seleção). Alguns navegadores apresentam os itens localizados entre [TD]<select>[TN] e [TD]</select>[TN] em um menu suspenso.

Então, já vimos dois dos tipos internos correspondentes à [TD]<input>[TN]: *radio* e *checkbox*. De fato, também já vimos um terceiro tipo: *text*. Como esse tipo é o padrão, não incluímos o parâmetro *type* = *text*, mas poderíamos ter feito isso. Dois outros tipos são *password* e *textarea*. Uma caixa *password* é igual a uma caixa *text*, exceto pelo fato de que os caracteres não são exibidos enquanto são digitados. Uma caixa *textarea* também é igual a uma caixa *text*, exceto pelo fato de poder conter várias linhas.

Voltando ao exemplo da Figura 7.29, nos deparamos com um exemplo de botão *submit* que, ao ser pressionado, faz com que as informações do usuário sejam enviadas de volta à máquina que forneceu o formulário. A exemplo de todos os outros tipos, *submit* é uma palavra reservada que o navegador compreende. O string *value* aqui é o rótulo atribuído ao botão e é exibido no formulário. Todas as caixas podem ter valores, mas aqui só precisamos desse recurso. Para as caixas *text*, o conteúdo do campo *value* é apresentado junto com o formulário, mas o usuário pode editar ou apagar esse conteúdo. As caixas *checkbox* e *radio* também podem ser inicializadas, mas com um campo *checked* (pois o campo *value* só apresenta o texto e não indica uma escolha).

Quando o usuário clica sobre o botão *submit*, o navegador reúne as informações coletadas em uma única linha longa e a envia de volta ao servidor para processamento. O caractere & é usado para separar campos, e o caractere + é usado para representar espaço. Em nosso exemplo de formulário, a linha seria semelhante ao conteúdo da Figura 7.30 (dividido em três linhas, porque a página do livro não tem largura suficiente):

[arte: ver original da p. 638]

[TD]

customer=John+Doe&address=100+Main+St.&city=White+Plains&

state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&

product=cheap&express=on [TN]

[F]Figura 7.30

[FL] Uma resposta possível do navegador ao servidor com informações fornecidas pelo usuário

O string seria transmitido ao servidor como uma única linha, e não três. Se uma caixa do tipo *checkbox* não for selecionada, ela será omitida do string. É função do servidor interpretar o string. Descreveremos mais adiante neste capítulo como isso poderia ser feito.

[T4] XML e XSL

A linguagem HTML, com ou sem formulários, não fornece qualquer estrutura para as páginas da Web. Ela também mistura o conteúdo com a formatação. À medida que o e-commerce e outras aplicações se tornam mais comuns, há uma necessidade crescente de estruturar páginas da Web e de separar o conteúdo da formatação. Por exemplo, um programa que pesquisa a Web em busca do melhor preço para algum livro ou CD precisa analisar muitas páginas da Web procurando pelo título e preço do item. Com páginas da Web em HTML, é muito difícil um programa descobrir onde está o título e onde está o preço.

Por essa razão, o W3C desenvolveu um aperfeiçoamento para a HTML, a fim de permitir que as páginas da Web sejam estruturadas para processamento automatizado. Foram desenvolvidas duas novas linguagens para esse propósito. A primeira, chamada **XML (eXtensible Markup Language)** descreve o conteúdo da Web de uma forma estruturada: a segunda, denominada **XSL (eXtensible Style Language)** descreve a formatação de modo independente do conteúdo. Ambos

são tópicos grandes e complicados; assim, nossa breve introdução a seguir irá apenas fornecer informações superficiais, mas deverá lhe dar uma idéia de como essas linguagens funcionam.

Considere o exemplo de documento XML da Figura 7.31. Ele define uma estrutura chamada [TD]book_list[TN], que é uma lista de livros ([TD]books[TN]). Cada livro tem três campos, o título, o autor e o ano de publicação. Essas estruturas são extremamente simples. São possíveis estruturas com campos repetidos (por exemplo, vários autores), campos opcionais (por exemplo, o título do CD-ROM incluído) e campos alternativos (por exemplo, o URL de uma livraria se o livro ainda estiver em catálogo, ou o URL de um site de leilões, se ele estiver esgotado).

Nesse exemplo, cada um dos três campos é uma entidade indivisível, mas os campos também podem ser subdivididos. Por exemplo, o campo de autor poderia ter sido criado da maneira a seguir, para oferecer um controle mais preciso sobre a pesquisa e a formatação:

[TD]

```
<author>
  <firsLname> Andrew </firsLname>
  <lasLname> Tanenbaum </iasLname>
</author> [TN]
```

Cada campo pode ser subdividido em subcampos e subsubcampos com profundidade arbitrária.

Tudo que o arquivo da Figura 7.31 faz é definir uma lista de livros contendo três livros. Ele nada nos diz sobre como exibir a página da Web na tela. Para fornecer as informações de formatação, precisamos de um segundo arquivo, *book_list.xsl*, contendo a definição de XSL. Esse arquivo é uma folha de estilos que informa como exibir a página. (Existem alternativas para folhas de estilos, como um

método para converter XML em HTML, mas essas alternativas estão além do escopo deste livro.)

Uma amostra de arquivo XSL para formatação da Figura 7.31 é apresentada na Figura 7.32. Depois de algumas declarações necessárias, incluindo o URL do padrão XSL, o arquivo conterá tags que começam com [TD]<html>[TN] e [TD]<body>[TN]. Essas tags definem o começo da página da Web, como de hábito. Em seguida, há uma definição de tabela, incluindo os cabeçalhos correspondentes às três colunas. Observe que além das tags [TD]<th>[TN] também existem as tags [TD]</th>[TN], algo com que não nos preocupamos até agora. As especificações de XML e XSL são muito mais rígidas que a especificação de HTML. Elas estabelecem que é obrigatório rejeitar sintaticamente arquivos incorretos, mesmo que o navegador possa descobrir o que o projetista da Web pretendia. Um navegador que aceita um arquivo XML ou XSL sintaticamente incorreto e repara ele próprio os erros não obedece ao padrão e será rejeitado em um teste de compatibilidade. No entanto, os navegadores têm permissão para indicar o erro. Essa medida um tanto draconiana é necessária para lidar com o imenso número de páginas da Web mal feitas que existem nos dias de hoje.

[arte: ver original da p. 640]

[TD]

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>
<book_list>
<book>
    <title> Computer Networks, 4/e </title>
    <author> Andrew S. Tanenbaum </author>
    <year> 2003 </year>
</book>
```

```
<book>
   <title> Modern Operating Systems, 2/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 2001 </year>
</book>
<book>
   <title> Structured Computer Organization, 4/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 1999 </year>
</book>
</book_list> [TN]
```

[F]Figura 7.31

[FL] Uma página da Web simples em XML

A declaração:

[TD] <xsl :for-each select="book_list/book"> [TN]

é análoga a uma instrução [TD]for[TN] em C. Ela faz o navegador iteragir pelo corpo do loop (que termina em [TD]<xsl:for-each>[TN]), uma iteração para cada [TD]book[TN]. Cada iteração fornece como saída cinco linhas: [TD]<tr>[TN], o título, o autor e ano, e ainda [TD]</tr>[TN]. Depois do loop, as tags de fechamento [TD]</body>[TN] e [TD]</html>[TN] são produzidas. O resultado da interpretação dessa folha de estilos pelo navegador é igual ao que haveria se a página da Web contivesse a tabela em linha. Porém, nesse formato, os programas podem, por exemplo, analisar o arquivo XML e encontrar com facilidade livros publicados depois de 2000. Vale a pena enfatizar que, embora nosso arquivo XSL contivesse uma espécie de loop, páginas da Web em XML e XSL ainda são estáticas, pois contêm apenas instruções para o navegador sobre como exibir a

página, exatamente como as páginas HTML. É claro que, para usar XML e XSL, o navegador tem de ser capaz de interpretar as linguagens XML e XSL, mas a maioria deles já tem essa capacidade. Não está claro se a XSL assumirá o lugar das folhas de estilos tradicionais.

[arte: ver original da p. 641]

[TD]

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/">
<html>
<body>
<table border="2">
  <tr>
     <th> Title</th>
     <th> Author</th>
     <th> Year </th>
  </tr>

  <xsl:for-each select="book_list/book">
  <tr>
     <td> <xsl:value-of select="title"/> </td>
     <td> <xsl:value-of select="author"/> </td>
     <td> <xsl:value-of select="year"/> </td>
 </tr>
 </xsl:for-each>
</table>
```

</body>

</html>

</xsl:template>

</xsl:stylesheet> [TN]

[F]Figura 7.32

[FL] Uma folha de estilos em XSL

Não mostramos como fazer isso, mas a linguagem XML permite ao projetista de Web sites compor arquivos de definição, nos quais as estruturas são definidas com antecedência. Esses arquivos de definição podem ser incluídos, tornando possível usá-los para construir páginas da Web complexas. Para obter mais informações sobre esse e muitos outros recursos de XML e XSL, consulte um dos muitos livros sobre o tema. Dois exemplos são (Livingston, 2002; e Williamson, 2001).

Antes de concluirmos nossa descrição de XML e XSL, vale a pena comentar uma batalha ideológica travada dentro do consórcio da WWW e da comunidade de projetistas da Web. O objetivo original da HTML era especificar a *estrutura* do documento, e não sua *aparência*. Por exemplo:

[TD] <h1> Deborah's Photos </h1> [TN]

instrui o navegador a enfatizar o cabeçalho, mas não diz nada sobre o tipo de fonte, o tamanho em pontos ou a cor da fonte. Isso foi deixado a cargo do navegador, que conhece as propriedades da tela de vídeo (por exemplo, quantos pixels ela tem). Porém, muitos projetistas de páginas da Web queriam ter controle absoluto sobre a aparência de suas páginas, e assim novas tags foram acrescentadas à HTML para controlar a aparência, como:

[TD] Deborah's Photos

[TN]

Além disso, foram acrescentados meios para controlar com precisão o posicionamento na tela. A dificuldade dessa abordagem é o fato dela não ser portátil. Embora uma página possa ser exibida perfeitamente com o navegador no qual ela foi desenvolvida, com outro navegador ou outra versão do mesmo navegador ou ainda com uma resolução de tela diferente, ela pode se transformar em uma completa desordem. Em parte, a XML foi uma tentativa de voltar ao objetivo original de especificar apenas a estrutura, e não a aparência de um documento. Porém, a XSL também é fornecida para administrar a aparência. Apesar disso, podem ocorrer abusos em ambos os formatos. Temos de contar com essa possibilidade.

A XML pode ser usada para outros fins além de descrever páginas da Web. Um uso crescente da XML é como uma linguagem para comunicação entre programas aplicativos. Em particular, o **SOAP (Simple Object Access Protocol)** é um modo de executar RPCs entre aplicações de forma independente da linguagem e do sistema. O cliente elabora a solicitação como uma mensagem XML e a envia ao servidor, usando o protocolo HTTP (descrito a seguir). O servidor devolve uma resposta como um mensagem XML formatada. Desse modo, as aplicações em plataformas heterogêneas podem se comunicar.

[T4] XHTML — eXtended HyperText Markup Language

A HTML continua evoluindo para atender a novas demandas. Muitas pessoas na indústria percebem que, no futuro, a maioria dos dispositivos capazes de reconhecer a Web não será formada por PCs, mas sim por dispositivos sem fios de tipo semelhante aos PDAs. Esses dispositivos têm memória limitada para os grandes navegadores repletos de heurísticas que tentam de algum modo lidar com páginas da Web sintaticamente incorretas. Desse modo, a próxima etapa após a HTML 4 é uma linguagem muito exigente. Ela é chamada **XHTML**

(eXtended HyperText Markup Language) em vez de HTML 5, porque é, em essência, a HTML 4 reformulada em XML. Com isso, queremos indicar que tags como [TD]<h1>[TN] não têm qualquer significado intrínseco. Para obter o efeito de HTML 4, é necessária uma definição no arquivo XSL. A XHTML é o novo padrão da Web e deve ser usada por todas as novas páginas da Web, até obter portabilidade máximo independente das plataformas e dos navegadores.

Existem seis diferenças principais e uma variedade de diferenças secundárias entre XHTML e HTML 4. Agora vamos examinar cuidadosamente as diferenças importantes. Primeiro, as páginas XHTML e os navegadores devem obedecer estritamente ao padrão. Nada de páginas da Web de qualidade inferior. Essa propriedade foi herdada da XML.

Em segundo lugar, todas as tags e atributos devem estar em letras minúsculas. Tags como [TD]<HTML>[TN] não são válidas em XHTML. O uso de tags como [TD]<html>[TN] agora é obrigatório. De modo semelhante, [TD]<img SRC="pic001.jpg">[TN] também é proibida, porque contém um atributo em letras maiúsculas.

Em terceiro lugar, as tags de fechamento são obrigatórias, até mesmo para [TD]</p>[TN]. Para tags que não têm nenhuma tag de fechamento natural, como [TD]
[TN], [TD]<hr>[TN] e [TD]<img>[TN], uma barra deve preceder o fechamento ">"; por exemplo:

[TD] [TN]

Em quarto lugar, os atributos devem estar contidos entre aspas. Por exemplo:

[TD] [TN]

não é mais permitido. O valor 500 tem de ficar entre aspas, como também o nome do arquivo JPEG, embora 500 seja apenas um número.

Em quinto lugar, as tags devem ficar aninhadas de maneira apropriada. No passado, o aninhamento não era exigido, desde que o estado final alcançado

fosse correto. Por exemplo:

[TD]<center> Vacation Pictures </center> [TN]

costumava ser válida. Em XHTML, essa declaração não é mais válida. As tags devem ser fechadas na ordem inversa em que foram abertas.

Em sexto lugar, todo documento deve especificar seu tipo de documento. Por exemplo, vimos essa propriedade na Figura 7.32. Para ver uma descrição de todas as mudanças, principais e secundárias, consulte *www.w3.org*.

[T3] 7.3.3 Documentos dinâmicos da Web

Até agora, o modelo que usamos é o da Figura 6.6: o cliente envia um nome de arquivo ao servidor, que então retorna o arquivo. Nos primeiros dias da Web, todo conteúdo era de fato estático como esse (apenas arquivos). Porém, nos últimos anos, o conteúdo se tornou cada vez mais dinâmico, isto é, gerado por demanda, em vez de ser armazenado em disco. A geração de conteúdo pode se dar no lado servidor ou no lado cliente. Vamos examinar agora cada um desses casos.

[T4] Geração dinâmica de páginas da Web do lado servidor

Para entender por que a geração de conteúdo do lado servidor é necessária, considere o uso de formulários, conforme descrevemos antes. Quando um usuário preenche um formulário e clica no botão *submit*, é enviada uma mensagem ao servidor, indicando que ela guarda o conteúdo de um formulário, juntamente com os campos que o usuário preencheu. Essa mensagem não é o nome de um arquivo a ser retornado. É preciso que a mensagem seja entregue a um programa ou script para processamento. Em geral, o processamento envolve a utilização das informações fornecidas pelo usuário pesquisar um registro em um banco de dados no disco do servidor e gerar uma página HTML personalizado que

será enviada de volta ao cliente. Por exemplo, em uma aplicação de e-commerce, depois que o usuário clica em *PROCEED TO CHECKOUT*, o navegador retorna o cookie com o conteúdo do carrinho de compras, mas algum programa ou script no servidor tem de ser invocado para processar o cookie e gerar uma página HTML em resposta. A página HTML pode exibir um formulário contendo a lista de itens no carrinho e o último endereço de entrega conhecido do usuário, juntamente com uma solicitação para verificar as informações e especificar o método de pagamento. As etapas exigidas para processar as informações de um formulário HTML estão ilustradas na Figura 7.33.

[arte: ver original p. 644]

[Dísticos]

[1] Usuário Navegador Servidor Script da CGI Banco de dados no disco

[2] 1 2 3 4

8 7 6 5

[3] 1. O usuário preenche o formulário

2. Formulário devolvido

3. Entregue à CGI

4. A CGI consulta o BD

5. Registro encontrado

6. A CGI constrói a página

7. Página retornada

8. Página exibida

[F]Figura 7.33

[FL] Etapas no processamento das informações de um formulário HTML

O caminho tradicional para lidar com formulários e outras páginas da Web

interativas é um sistema chamado **CGI (Comum Gateway Interface)**. A CGI é uma interface padronizada para permitir que servidores da Web se comuniquem com programas e scripts de back end que possam aceitar a entrada (por exemplo, de formulários) e gerar páginas HTML em resposta. Normalmente, esses back ends são scripts criados na linguagem de programação Perl, porque os scripts de Perl são mais fáceis e mais rápidos de gerar que os programas (pelo menos, se você souber programar em Perl). Por convenção, eles residem em um diretório chamado *cgi-bin*, visível no URL. Às vezes, é usada a linguagem de programação Python em lugar de Perl.

Um exemplo de como a CGI costuma funcionar é ilustrado por um produto da Truly Great Products Company que vem sem um cartão de registro de garantia. Em vez disso, o cliente deve ir até *www.tgpc.com* para se registrar on-line. Nessa página, existe um hiperlink com a inscrição:

[TD] Clique aqui para registrar seu produto [TN]

Esse link aponta para um script de Perl, digamos, *www.tgpc.com/cgi-bin/reg.perl*. Ao ser invocado sem parâmetros, esse script envia de volta uma página HTML contendo o formulário de registro. Quando o usuário preenche o formulário e clica em *submit*, é enviada de volta a esse script uma mensagem que contém os valores preenchidos com o uso do estilo da Figura 7. 30. Então, o script de Perl analisa os parâmetros, cria uma entrada no banco de dados para o novo cliente e envia de volta uma página HTML fornecendo um número de registro e um número de telefone da assistência técnica. Esse não é o único modo de lidar com formulários, mas é um modo comum. Existem muitos livros sobre a criação de scripts da CGI e programação em Perl. Alguns exemplos são (Hanegan, 2001; Lash, 2002; e Meltzer e Michalski, 2001).

Os scripts da CCI não são a única maneira de gerar conteúdo dinâmico no lado servidor. Outra forma comum é incorporar pequenos scripts em páginas HTML e

fazer com que eles sejam executados pelo próprio servidor para gerar a página. Uma linguagem popular para criação desses scripts é **PHP (PHP: Hypertext Preprocessor)**. Para usar essa linguagem, o servidor tem de reconhecer PHP (da mesma maneira que um navegador tem de reconhecer XML para interpretar páginas da Web escritas em XML). Normalmente, os servidores esperam que as páginas da Web contendo PHP tenham a extensão de arquivo *php*, em vez de *html* ou *htm*.

Um minúsculo script PHP está ilustrado na Figura 7.34; ele deve funcionar com qualquer servidor que tenha o PHP instalado. O script contém HTML normal, exceto pelo script PHP no interior da tag [TD]<?php ... ?>[TN]. Ele gera uma página da Web informando o que sabe sobre o navegador que o invocou. Em geral, os navegadores enviam algumas informações junto com suas solicitações (e quaisquer cookies aplicáveis), e essas informações são inseridas na variável *HTTP_USER_AGENT*. Quando essa listagem é inserida em um arquivo *test.php* no diretório WWW da empresa ABCD, a digitação do URL *www.abcd.com/test.php* produzirá uma página da Web informando ao usuário que navegador, linguagem e sistema operacional ele está usando.

[arte: ver original da p. 645]

```
[TD]
<html>
<body>

<h2> Isto é tudo que sei sobre você </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html> [TN]
```

[F]Figura 7.34

[FL] Um exemplo de página HTML com PHP incorporado

O PHP é especialmente indicado para manipulação de formulários e é mais simples que usar um script da CGI. Como exemplo de seu funcionamento com formulários, consulte a Figura 7.35(a). Essa figura contém uma página HTML normal com um formulário. O único item incomum é a primeira linha, que especifica que o arquivo *action.php* deve ser invocado para tratar os parâmetros, depois que o usuário tiver preenchido e enviado o formulário. A página exibe duas caixas de texto, uma com a solicitação de um nome, e a outra solicitando a idade. Depois que as duas caixas são preenchidas e o formulário é transmitido, o servidor analisa o string semelhante ao da Figura 7.30 que foi enviado, inserindo o nome na variável *name* e a idade na variável *age*. Em seguida, ele começa a processar o arquivo *action.php*, mostrado na Figura 7.35(b) como uma resposta. Durante o processamento desse arquivo, são executados os comandos de PHP. Se o usuário preencheu "Isabela" e "24" nas caixas, o arquivo HTML enviado de volta será o que está representado na Figura 7.35(c). Desse modo, o tratamento de formulários se torna extremamente simples com o uso de PHP.

Embora o PHP seja fácil de usar, na realidade ele é uma poderosa linguagem de programação orientada para formar a interface entre a Web e um banco de dados servidor. Ele tem variáveis, strings, arrays e a maioria das estruturas de controle encontradas em C, mas apresenta instruções de E/S muito mais eficientes que o simples uso de *printf*. O código-fonte do PHP é aberto e está disponível gratuitamente. Ele foi projetado de forma específica para funcionar bem com o Apache, que também tem um código-fonte aberto e é o servidor da Web mais utilizado em todo o mundo. Para obter mais informações sobre o PHP, consulte (Valade, 2002).

[arte: ver original da p. 646]

[TD]

<html>

<body>

<form action="action.php" method="post">

<p> Please enter your name: <input type="text" name="name"> </p>

<p> Please enter your age: <input type="text" name="age"> </p>

<input type="submit">

</form>

</body>

</html> [TN]

(a)

[TD]

<html>

<body>

<h1> Reply: </h1>

Hello <?php echo $name; ?>.

Prediction: next year you will be <?php echo $age + 1; ?>

</body>

</html> [TN]

(b)

[TD]

<html>

<body>

<h1> Reply: </h1>

Hello Isabela.

Prediction: next year you will be 25

</body>

</html> [TN]

(c)

[F]Figura 7.35

[FL] (a) Uma página da Web contendo um formulário. (b) Um script PHP para manipular a saída do formulário. (c) Saída do script PHP quando as entradas são "Isabela" e 24, respectivamente

Agora, vimos duas maneiras diferentes de gerar páginas dinâmicas de HTML: scripts da CGI e PHP incorporado. Existe ainda uma terceira técnica, chamada **JSP (JavaServer Pages)**, semelhante ao PHP, exceto pelo fato da parte dinâmica ser escrita na linguagem de programação Java, e não em PHP. As páginas que utilizam essa técnica têm a extensão de arquivo *jsp*. Uma quarta técnica, **ASP (Active Server Pages)**, é a versão da Microsoft para PHP e JavaServer Pages. Ela utiliza a linguagem de scripts patenteada da Microsoft, o Visual Basic Script, para gerar o conteúdo dinâmico. As páginas que empregam essa técnica têm a extensão *asp*. A escolha entre *PHP*, *JSP* e *ASP* normalmente está mais relacionada à política (código-fonte aberto *versus* Sun *versus* Microsoft) do que à tecnologia, pois as três linguagens são comparáveis em linhas gerais.

A coleção de tecnologias para gerar conteúdo durante a execução às vezes é chamada **HTML dinâmica.**

[T4] Geração dinâmica de páginas da Web do lado cliente

Os scripts CGI, PHP, JSP e ASP resolvem o problema de manipular formulários e interações com bancos de dados no servidor. Todos eles podem aceitar informações de entrada de formulários, pesquisar informações em um ou mais bancos de dados e gerar páginas HTML com os resultados. O que nenhum deles

pode fazer é responder a movimentos do mouse ou interagir diretamente com os usuários. Para esse propósito, é necessário ter scripts incorporados em páginas HTML executadas na máquina cliente e não na máquina servidora. A partir da HTML 4.0, esses scripts são permitidos com o uso da tag [TD]<script>[TN]. A linguagem de scripts mais popular para o lado cliente é o **JavaScript**; portanto, vamos examiná-lo rapidamente.

JavaScript é uma linguagem de scripts, *muito* livremente inspirada em algumas idéias da linguagem de programação Java. Definitivamente não se trata de Java. Como outras linguagens de scripts, essa é uma linguagem de nível muito alto. Por exemplo, em uma única linha de JavaScript é possível mostrar uma caixa de diálogo, aguardar a entrada de texto e armazenar o string resultante em uma variável. Características de alto nível como essa tornam o JavaScript ideal para projetar páginas da Web interativas. Por outro lado, o fato de não ser padronizada e de estar mudando com muita rapidez torna extremamente difícil escrever programas JavaScript que funcionem em todas as plataformas, mas talvez algum dia essa linguagem se estabilize.

Como exemplo de um programa em JavaScript, considere o código da Figura 7.36. Como o da Figura 7.35(a), ele exibe um formulário que pede o nome e a idade de uma pessoa, e depois prevê qual será a idade da pessoa no ano seguinte. O corpo é quase igual ao do exemplo de PHP; a principal diferença é a declaração do botão de envio e a instrução de atribuição que ele contém. Essa instrução de atribuição informa que o navegador deve invocar o script *response* quando houver um clique em um botão e repassá-lo ao formulário como um parâmetro.

A grande novidade aqui é a declaração da função do JavaScript *response* no início do arquivo HTML, uma área normalmente reservada para títulos, cores de fundo e assim por diante. Essa função extrai o valor do campo *name* do formulário e o

armazena na variável *person*, sob a forma de um string. Ele também extrai o valor do campo *age*, converte esse valor em um inteiro usando a função *eval*, soma 1 ao valor e armazena o resultado em *years*. Em seguida, ele abre um documento para saída, executa quatro operações de escrita usando o método *writeln* e fecha o documento. O documento é um arquivo HTML, como podemos observar pelas diversas tags de HTML que ele contém. O navegador exibe então o documento na tela.

É muito importante compreender que, embora a Figura 7.35 e a Figura 7.36 pareçam semelhantes, elas são processadas de forma totalmente diferente. Na Figura 7.35, depois que o usuário clica no botão *submit*, o navegador reúne as informações em um longo string no estilo da Figura 7.30 e o envia ao servidor que transmitiu a página. O servidor vê o nome do arquivo PHP e o executa. O script PHP produz uma nova página HTML, e essa página é enviada de volta ao navegador para exibição. No caso da Figura 7.36, quando o botão *submit* é acionado, o navegador interpreta uma função do JavaScript contida na página. Todo o trabalho é feito no local, dentro do navegador. Não há nenhum contato com o servidor. Em conseqüência disso, o resultado é exibido quase instantaneamente; no caso do PHP, pode haver um retardo de vários segundos antes do código HTML resultante chegar ao cliente. A diferença entre a criação de scripts no lado servidor e a criação de scripts no lado cliente é ilustrado na Figura 7.37, incluindo as etapas envolvidas. Em ambos os casos, as etapas numeradas começam depois que o formulário é exibido. A etapa 1 consiste na aceitação da entrada do usuário. Em seguida, vem o processamento da entrada, diferente nos dois casos.

[arte: ver original da p. 648]

[TD]

<html>

```
<head>

<script language="javascript" type="text/javascript">

function response(test_form) {

   var person = test_form.name.value;

   var years = eval(test_form.age.value) + 1;

   document.open();

   document.writeln("<html> <body>");

   document.writeln("Hello " + person + ".<br>");

   document.writeln("Prediction: next year you will be " + years + ".");

   document.writeln("</body> </html>");

   document.close();

}

</script>

</head>

<body>

<form>

Please enter your name: <input type="text" name="name">

<p>

Please enter your age: <input type="text" name="age">

<p>

<input type="button" value="submit" onclick="response(this.form)">

</form>

</body>

</html> [TN]
```

[F]Figura 7.36

[FL] Utilização do JavaScript para processar um formulário

Essa diferença não significa que o JavaScript seja melhor que o PHP. Seus usos são completamente diferentes. O PHP (e, por implicação, JSP e ASP) são usados quando é necessária a interação com um banco de dados remoto. O JavaScript é utilizado quando a interação se dá com o usuário no computador cliente. Sem dúvida é possível (e comum) haver páginas de HTML que utilizam o PHP e o JavaScript, embora eles não possam realizar o mesmo trabalho ou ter o mesmo botão.

[arte: ver original p. 649a]

[Dísticos]

[1] Usuário Navegador Servidor

1 2

4 3

(a) Módulo de PHP

[2] Usuário Navegador Servidor

1

2

JavaScript (b)

[F]Figura 7.37

[FL] (a) Geração de script do lado servidor com o PHP. (b) Geração de script do lado cliente com o JavaScript

O JavaScript é uma linguagem de programação completa, com toda a capacidade de C ou Java. Ele tem variáveis, strings, arrays, objetos, funções e todas as estruturas de controle habituais. O JavaScript também tem um grande número de recursos específicos para as páginas da Web, inclusive a habilidade para gerenciar janelas e quadros, definir e obter cookies, lidar com formulários e manipular hiperlinks. Um exemplo de um programa JavaScript que utiliza uma

função recursiva é dado na Figura 7.38.

[arte: ver original da p. 649b]

[TD]

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   function factorial(n) {if (n == 0) return 1; else return n * factorial(n - 1);}
   var r = eval(test_form.number.value);       //  r = argumento digitado
   document.myform.mytext.value = "Here are the results.\n";
   for (var i = 1; i <= r; i++)      // imprime uma linha de 1 até r
      document.myform.mytext.value += (i + "! =  " + factorial(i) + "\n");
}
</script>
</head>
<body>
<form name="myform">
Please enter a number: <input type="text" name="number">
<input type="button" value="compute table of factorials"
onclick="response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html> [TN]
```

[F]Figura 7.38

[FL] Um programa em JavaScript para calcular e imprimir fatoriais

O JavaScript também pode acompanhar a movimentação do mouse sobre objetos na tela. Em muitas páginas da Web com JavaScript acontece algo quando o cursor do mouse é deslocado sobre algum texto ou imagem. Com freqüência, a imagem muda ou um menu aparece de repente. É fácil programar esse tipo de comportamento em JavaScript, e ele resulta em páginas da Web bastante ativas. Um exemplo é mostrado na Figura 7.39.

[arte: ver original da p. 650]

[TD]

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/~ast/im/kitten.jpg";
document.myurl[1] = "http://www.cs.vu.nl/~ast/im/puppy.jpg";
document.myurl[2] = "http://www.cs.vu.nl/~ast/im/bunny.jpg";
function pop(m) {
   var urx = "http://www.cs.vu.nl/~ast/im/cat.jpg";
   popupwin =
window.open(document.myurl[m],"mywind","width=250,height=250");
}
</script>
</head>
<body>
<p> <a href="#" onMouseover="pop(0); return false;" > Kitten  </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;" > Puppy  </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;" > Bunny  </a> </p>
```

</body>

</html> [TN]

[F]Figura 7.39

[FL] Uma página da Web interativa que responde ao movimento do mouse

O JavaScript não é a única maneira de tornar as páginas da Web altamente interativas. Outro método popular é usar **miniaplicativos (applets).** Os miniaplicativos são pequenos programas em Java que foram compilados em instruções de máquina para um computador virtual chamado **JVM (Java Virtual Machine).** Os miniaplicativos podem ser incorporados em páginas de HTML (entre as tags [TD]<applet>[TN] e [TD]</applet>[TN]) e interpretados por navegadores compatíveis com a JVM. Pelo fato dos miniaplicativos Java serem interpretados em vez de serem executados diretamente, o interpretador Java pode evitar que eles se comportem mal, pelo menos na teoria. Na prática, os criadores de miniaplicativos descobriram um fluxo quase infinito de bugs a serem explorados nas bibliotecas de E/S do Java.

A resposta da Microsoft aos miniaplicativos Java da Sun foi permitir que as páginas da Web contivessem **controles ActiveX**, que são programas compilados na linguagem de máquina do Pentium e executados no hardware bruto. Esse recurso os torna imensamente mais rápidos e mais flexíveis que os miniaplicativos Java interpretados, porque eles podem fazer quase tudo que um programa é capaz de realizar. Quando o Internet Explorer detecta um controle ActiveX em uma página da Web, ele o transfere por download, verifica sua identidade e o executa. Porém, a transferência e a execução de programas externos faz surgirem questões de segurança, como veremos no Capítulo. 8.

Tendo em vista que quase todos os navegadores podem interpretar tanto programas em Java quanto em JavaScript, um projetista que queira criar uma

página da Web altamente interativa tem a opção de usar pelo menos duas técnicas e, se a portabilidade para várias plataformas não for um problema, ele ainda terá o ActiveX à sua disposição. Como regra geral, os programas JavaScript são mais fáceis de escrever, os miniaplicativos Java são executados com mais rapidez, e os controles ActiveX são os mais rápidos de todos. Além disso, como todos os navegadores implementam exatamente a mesma JVM, mas não há dois navegadores que implementem a mesma versão de JavaScript, os miniaplicativos Java são mais portáteis que os programas em JavaScript. Para obter mais informações sobre o JavaScript, existem muitos livros, cada um com muitas (freqüentemente mais de 1000) páginas. Alguns exemplos são (Easttom, 2001; Harris, 2001; e McFedries, 2001).

Antes de deixarmos o assunto de conteúdo dinâmico da Web, vamos resumir brevemente o que estudamos até agora. Páginas da Web completas podem ser geradas durante a execução por diversos scripts na máquina servidora. Uma vez recebidas pelo navegador, elas são tratados como páginas normais de HTML e são simplesmente exibidas. Os scripts podem ser escritos em Perl, PHP, JSP ou ASP, como mostra a Figura 7.40.

[arte: ver original p. 651]

[Dísticos]

[1] Interpretador XSL

Interpretador XML

Interpretador HTML

Interpretador JavaScript

[2] Máquina cliente

Navegador

Auxiliar

Plug-in

[3] Máquina servidora

Servidor

PHP

JSP

ASP Script da CGI

[F]Figura 7.40

[FL] As diversas maneiras de gerar e exibir conteúdo

A geração de conteúdo dinâmico também é possível no lado cliente. As páginas da Web podem ser escritas em XML e depois convertidas em HTML, de acordo com um arquivo XSL. Os programas JavaScript podem executar cálculos arbitrários. Finalmente, plug-ins e aplicações auxiliares podem ser usados para exibir conteúdo em uma variedade de formatos.

[T3] 7.3.4 HTTP — HyperText Transfer Protocol

O protocolo de transferência utilizado em toda a World Wide Web é o **HTTP (HyperText Transfer Protocol)**. Ele especifica as mensagens que os clientes podem enviar aos servidores e que respostas eles receberão. Cada interação consiste em uma solicitação ASCII, seguida por uma resposta RFC 822 semelhante ao MIME. Todos os clientes e todos os servidores devem obedecer a esse protocolo. Ele é definido na RFC 2616. Nesta seção, estudaremos algumas de suas propriedades mais importantes.

[T4] Conexões

O modo habitual de um navegador entrar em contato com um servidor é estabelecer uma conexão TCP para a porta 80 da máquina servidora, embora esse procedimento não seja exigido formalmente. O vantagem de se usar o TCP é que

nem os navegadores nem os servidores têm de se preocupar com mensagens perdidas, mensagens duplicadas, mensagens longas ou confirmações. Todos esses assuntos são tratados pela implementação do TCP.

No HTTP 1.0, depois que a conexão era estabelecida, uma única solicitação era enviada e uma única resposta era devolvida. Então, a conexão TCP era encerrada. Em um mundo no qual as páginas da Web típicas consistiam inteiramente em texto HTML, esse método era adequado. Após alguns anos, a página da Web média continha grandes números de ícones, imagens e outros atrativos visuais, e assim o estabelecimento de uma conexão TCP para transportar um único ícone se tornou um modo de operação muito dispendioso.

Essa observação levou ao lançamento do HTTP 1.1, que admite **conexões persistentes.** Com elas, é possível estabelecer uma conexão TCP, enviar uma solicitação e obter uma resposta, e depois enviar solicitações adicionais e receber respostas adicionais. Amortizando o custo da instalação e da liberação do TCP por várias solicitações, o overhead relativo devido ao TCP é muito menor por solicitação. Também é possível transportar as solicitações por pipeline, ou seja, enviar a solicitação 2 antes de chegar a resposta à solicitação 1.

[T4] Métodos

Embora o HTTP tenha sido projetado para utilização na Web, ele foi criado de modo mais geral que o necessário, visando às futuras aplicações orientadas a objetos. Por essa razão, são aceitas operações chamadas **métodos**, diferentes da simples solicitação de uma página da Web. Essa generalidade permitiu que o SOAP viesse a existir. Cada solicitação consiste em uma ou mais linhas de texto ASCII, sendo a primeira palavra da primeira linha o nome do método solicitado. Os métodos internos estão listados na Figura 7.41. Para acessar objetos gerais, também podem estar disponíveis métodos adicionais específicos de objetos. Os

nomes diferenciam letras maiúsculas de minúsculas; portanto, *GET* é um método válido, mas *get* não é.

O método *GET* solicita ao servidor que envie a página (ou objeto, no caso mais genérico; na prática, apenas um arquivo). A página é codificada em MIME de forma adequada. A grande maioria das solicitações a servidores da Web tem a forma de métodos *GET*. A forma usual de *GET* é:

[TD]GET nomearq HTTP/1.1[TN]

onde *nomearq* identifica o recurso (arquivo) a ser buscado e 1.1 é a versão do protocolo que está sendo usado.

O método *HEAD* solicita apenas o cabeçalho da mensagem, sem a página propriamente dita. Esse método pode ser usado para se obter a data da última modificação feita na página, para reunir informações destinadas à indexação, ou apenas para testar a validade de um URL.

[arte: ver original p. 653]

[T]Tabela

Método	**Descrição**
GET	Solicita a leitura de uma página da Web
HEAD	Solicita a leitura de um cabeçalho de página da Web
PUT	Solicita o armazenamento de uma página da Web
POST	Acrescenta a um recurso (por exemplo, uma página da Web)
DELETE	Remove a página da Web
TRACE	Ecoa a solicitação recebida
CONNECT	Reservado para uso futuro
OPTIONS	Consulta certas opções

[F]Figura 7.41

[FL] Os métodos internos de solicitações HTTP

O método *PUT* é o inverso de *GET*: em vez de ler, ele grava a página. Esse método possibilita a criação de um conjunto de páginas da Web em um servidor remoto. O corpo da solicitação contém a página. Ela pode ser codificada com o uso de MIME. Nesse caso, as linhas após *PUT* podem incluir *Content-Type* e cabeçalho de autenticação, para demonstrar que o chamador de fato tem permissão para executar a operação solicitada.

Um pouco semelhante a *PUT* é o método *POST*, que também transporta um URL. No entanto, em vez de substituir os dados existentes, os novos dados são "anexados" a ele, em um sentido mais genérico. Enviar uma mensagem a um newsgroup ou incluir um arquivo em um BBS são dois exemplos de anexação nesse contexto. Na prática, nem *PUT* nem *POST* são muito utilizados hoje.

DELETE faz exatamente isso: exclui a página. A exemplo de *PUT*, a permissão e a autenticação têm papel fundamental. Não há garantia de que *DELETE* tenha sido bem-sucedido pois, mesmo que o servidor HTTP remoto esteja pronto para excluir a página, o arquivo subjacente pode ter um modo que impeça o servidor HTTP de modificá-lo ou excluí-lo.

O método *TRACE* serve para depuração. ele instrui o servidor a enviar de volta a solicitação. Esse método é útil quando as solicitações não estão sendo processadas corretamente e o cliente deseja saber qual solicitação o servidor recebeu de fato.

O método *CONNECT* não é usado atualmente. Ele é reservado para uso futuro.

O método *OPTIONS* fornece um meio para que o cliente consulte o servidor sobre suas propriedades ou sobre as de um arquivo específico.

Toda solicitação obtém uma resposta que consiste em uma linha de status e, possivelmente, informações adicionais (por exemplo, uma página da Web ou parte dela). A linha de status contém um código de status de três dígitos informando se a solicitação foi atendida e, se não foi, por que não. O primeiro

dígito é usado para dividir as respostas em cinco grupos importantes, como mostra a Figura 7.42. Os códigos 1xx raramente são usados na prática. Os códigos 2xx significam que a solicitação foi tratada com sucesso, e que o conteúdo (se houver) está sendo retornado. Os códigos 3xx informam ao cliente que ele deve procurar em outro lugar, usando um URL diferente ou seu próprio cache (conforme descreveremos mais adiante). Os códigos 4xx significam que a solicitação falhou devido a um erro do cliente, como uma solicitação inválida ou uma página inexistente. Finalmente, os erros 5xx significam que o próprio servidor tem um problema, seja causado por um erro em seu código ou por uma sobrecarga temporária.

[arte: ver original p. 654]

[T]Tabela

Código	**Significado**	**Exemplos**
1xx	Informação	100 = server agrees to handle client's request
2xx	Sucesso	200 = request succeeded; 204 = no content present
3xx	Redirecionamento	301 = page moved; 304 = cached page still valid
4xx	Erro do cliente	403 = forbidden page; 404 = page can not found
5xx	Erro do servidor	500 = internal server error; 503 = try again later

[F]Figura 7.42

[FL] Os grupos de respostas de código de status

[T4] Cabeçalhos de mensagens

A linha de solicitação (por exemplo, a linha com o método *GET*) pode ser seguida por linhas adicionais com mais informações. Elas são chamadas **cabeçalhos de solicitação.** Essas informações podem ser comparadas aos parâmetros de uma chamada de procedimento. As respostas também podem ter **cabeçalhos de**

resposta. Alguns cabeçalhos podem ser usados em um ou outro sentido. Uma seleção dos mais importantes é dada na Figura 7.43.

O cabeçalho *User-Agent* permite ao cliente informar o servidor sobre seu navegador, sistema operacional e outras propriedades. Na Figura 7.34, vimos que o servidor obtinha essas informações por página e podia produzi-las por demanda em um script PHP. Esse cabeçalho é usado pelo cliente para munir o servidor com as informações.

Os quatro cabeçalhos *Accept* informam ao servidor o que o cliente está disposto a aceitar na eventualidade de ele ter um repertório limitado daquilo que é aceitável. O primeiro cabeçalho especifica os tipos MIME que são bem-vindos (por exemplo, text/html). O segundo fornece o conjunto de caracteres (por exemplo, ISO-8859-5 ou Unicode-1-1). O terceiro lida com métodos de compactação (por exemplo, gzip). O quarto indica um idioma natural (por exemplo, espanhol). Se o servidor tiver uma opção de páginas, ele poderá usar essas informações para fornecer o que o cliente está procurando. Se ele for incapaz de satisfazer à solicitação, será retornado um código de erro e a solicitação falhará.

O cabeçalho *Host* identifica o servidor. Ele é retirado do URL. Esse cabeçalho é obrigatório, e é usado porque alguns endereços IP podem servir vários nomes DNS, e o servidor precisa ter algum meio de identificar o host a quem deve entregar a solicitação.

O cabeçalho *Authorization* é necessário para páginas protegidas. Nesse caso, o cliente talvez tenha de provar que tem direito de ver a página solicitada. Esse cabeçalho é usado para esse caso específico.

Embora os cookies sejam tratados na RFC 2109 e não na RFC 2616, eles também têm dois cabeçalhos. O cabeçalho *Cookie* é usado por clientes para retornar ao servidor um cookie enviado anteriormente por alguma máquina no domínio do servidor.

[arte: ver original p. 655]

[T]Tabela

Cabeçalho	Tipo	Conteúdo
User-Agent	Solicitação	Informações sobre o navegador e sua plataforma
Accept	Solicitação	O tipo de páginas o cliente pode manipular
Accept-Charset	Solicitação	Os conjuntos de caracteres aceitáveis para o cliente
Accept-Encoding	Solicitação	As codificações de páginas que o cliente pode manipular
Accept-Language	Solicitação	Os idiomas com os quais o cliente pode lidar
Host	Solicitação	O nome DNS do servidor
Authorization	Solicitação	Uma lista das credenciais do cliente
Cookie	Solicitação	Envia um cookie definido anteriormente de volta ao servidor
Date	Ambos	Data e hora em que a mensagem foi enviada
Upgrade	Ambos	O protocolo para o qual transmissor deseja alternar
Server	Resposta	Informações sobre o servidor
Content-Encoding	Resposta	Como o conteúdo está codificado (por exemplo, gzip)
Content-Language	Resposta	O idioma usado na página
Content-Length	Resposta	O comprimento da página em bytes
Content-Type	Resposta	O tipo MIME da página
Last-Modified	Resposta	Data e hora da última modificação na página
Location	Resposta	Um comando para o cliente enviar sua solicitação a outro lugar
Accept-Ranges	Resposta	O servidor aceitará solicitações de intervalos de

bytes

Set-Cookie Resposta O servidor deseja que o cliente grave um cookie

[F]Figura 7.43

[FL] Alguns cabeçalhos de mensagens HTTP

O cabeçalho *Date* pode ser usado em ambos os sentidos e contém a hora e a data em que a mensagem foi enviada. O cabeçalho *Upgrade* é usado para facilitar a transição para uma versão futura (possivelmente incompatível) do protocolo HTTP. Ele permite ao cliente anunciar o que pode admitir e permite ao servidor declarar o que está usando.

Agora, vamos aos cabeçalhos usados exclusivamente pelo servidor em resposta a solicitações. O primeiro, *Server*, permite ao servidor saber quem ele e conhecer algumas de suas propriedades, se desejar.

Os quatro cabeçalhos seguintes, todos começando com *Content-*, permitem ao servidor descrever propriedades da página que está enviando.

O Cabeçalho *Last-Modified* informa quando a página foi modificada pela última vez. Esse cabeçalho desempenha uma função importante no armazenamento de páginas no cache.

O cabeçalho *Location* é usado pelo servidor para informar ao cliente que ele deve tentar outro URL. Esse cabeçalho pode ser usado se a página tiver sido deslocada ou para permitir que vários URLs se refiram à mesma página (possivelmente em servidores distintos). Ele também é usado por empresas que têm uma página da Web principal no domínio *com*, mas que redirecionam os clientes para uma página nacional ou regional de acordo com seu endereço IP ou com seu idioma preferido.

Se uma página for muito grande, um pequeno cliente talvez não queira recebê-la toda de uma vez. Alguns servidores aceitarão solicitações de intervalos de bytes,

de forma que a página possa ser obtida em várias unidades pequenas. O cabeçalho *Accept-Ranges* anuncia a disposição do servidor para lidar com esse tipo de solicitação de páginas parciais.

O segundo cabeçalho de cookie, *Set-Cookie*, é a forma como os servidores enviam cookies aos clientes. Espera-se que o cliente grave o cookie e o devolva em solicitações subseqüentes ao servidor.

[T4] Exemplo e utilização do HTTP

Como o HTTP é um protocolo ASCII, é muito fácil a comunicação direta entre uma pessoa em um terminal (diferente de um navegador) e servidores da Web. Basta uma conexão TCP para a porta 80 no servidor. Os leitores devem experimentar pessoalmente essa situação (de preferência usando um sistema UNIX, porque alguns outros sistemas não retornam o status da conexão). Use a seqüência de comandos a seguir:

[TD]

telnet www.ietf.org 80 >log

GET /rfc.html HTTP/1.1

Host: www.ietf.org

close [TN]

Essa seqüência de comandos inicia uma conexão telnet (isto é, TCP) para a porta 80 no servidor da Web da IETF, *www.ietf.org*. O resultado da sessão é redirecionado para o arquivo *log*, a fim de ser inspecionado mais tarde. Em seguida, vem o comando *GET* que identifica o arquivo e o protocolo. A próxima linha é o cabeçalho *Host* obrigatório. A linha em branco também é necessária. Ela indica ao servidor que não existem mais cabeçalhos de solicitação. O comando *close* instrui o programa telnet a interromper a conexão.

O *log* pode ser inspecionado com o uso de qualquer editor. Ele deve começar de maneira semelhante à listagem da Figura 7.44, a menos que a IETF tenha feito alguma alteração recente.

As três primeiras linhas são a saída do programa telnet, e não do site remoto. A linha que inicializa o HTTP/1.1 é a resposta da IETF, informando que está disposta a se comunicar com você usando HTTP/1.1. Em seguida, há uma série de cabeçalhos, e depois o conteúdo. Já vimos todos os cabeçalhos, com exceção de *ETag*, um identificador exclusivo de página relacionado ao armazenamento no cache, e de *X-Pad*, um cabeçalho não padronizado e talvez um artifício para contornar bugs em algum navegador.

[T3] 7.3.5 Aperfeiçoamentos de desempenho

A popularidade da Web quase a arruinou. Servidores, roteadores e linhas freqüentemente estão sobrecarregadas. Muitas pessoas começaram a chamar a WWW a World Wide Wait (grande espera mundial). Como conseqüência desses infindáveis retardos, os pesquisadores desenvolveram várias técnicas para melhorar o desempenho. Examinaremos agora três delas: caches, replicação de servidores e redes de entrega de conteúdo.

[arte: ver original da p. 657]

[TD]

Trying 4.17.168.6...

Connected to www.ietf.org.

Escape character is '^]'.

HTTP/1.1 200 OK

Date: Wed, 08 May 2002 22:54:22 GMT

Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a

Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT

```
ETag: "2a79d-c8b-39bce48d"

Accept-Ranges: bytes

Content-Length: 3211

Content-Type: text/html

X-Pad: avoid browser bug


<html>

<head>

<title>IETF RFC Page</title>


<script language="javascript">

function url() {

var x = document.form1.number.value

if (x.length == 1) {x = "000" + x }

if (x.length == 2) {x = "00" + x }

if (x.length == 3) {x = "0" + x }

document.form1.action = "/rfc/rfc" + x + ".txt"

document.form1.submit

}

</script>


</head> [TN]
```

[F]Figura 7.44

[FL] O início da saída de *www.ietf.org/rfc.html*

[T4] Armazenamento em cache

Um modo bastante simples de melhorar o desempenho é gravar páginas que

foram solicitadas, caso elas tenham de ser usadas novamente. Essa técnica é especialmente efetiva com páginas muito visitadas, como *www.yahoo.com* e *www.cnn.com*. Guardar páginas para uso subseqüente é uma técnica chamada **armazenamento em cache.** O procedimento habitual é algum processo, chamado **proxy**, manter o cache. Para usar o armazenamento em cache, um navegador pode ser configurado para fazer todas as solicitações de páginas a um proxy, em vez de solicitá-las ao servidor real das páginas. Se o proxy tiver a página, ele a retornará imediatamente. Se não tiver, ele buscará a página no servidor, guardando-a no cache para uso futuro e retornando a página para o cliente que a solicitou.

Duas questões importantes relacionadas ao armazenamento em cache são:

1. Quem deve realizar o armazenamento em cache?
2. Por quanto tempo as páginas devem ficar armazenadas no cache?

Há várias respostas para a primeira pergunta. Os PCs individuais freqüentemente executam proxies para que eles possam pesquisar com rapidez páginas já visitadas. Em uma LAN de empresa, o proxy é quase sempre uma máquina compartilhada por todas as máquinas da LAN; assim, se um usuário que examina uma certa página e depois outra na mesma LAN quiser voltar à página anterior, ela poderá ser buscada no cache do proxy. Muitos ISPs também utilizam proxies, a fim de acelerar o acesso de todos os seus clientes. Com freqüência, todos esses caches operam ao mesmo tempo, e assim as solicitações vão primeiro para o proxy local. Se isso falhar, o proxy local consultará o proxy da LAN. Se não der certo, o proxy da LAN tentará o proxy do ISP. Esse último deve ter sucesso, seja obtendo a página em seu cache, em um cache de nível mais alto ou no próprio servidor. Um esquema envolvendo vários caches acessados em seqüência é chamado **armazenamento em cache hierárquico.** Uma implementação possível é ilustrada na Figura 7.45.

[arte: ver original p. 658]

[Dísticos]

[1] Proxy

Máquina cliente LAN do lado cliente

[2] Internet Proxy

LAN do ISP

[F]Figura 7.45

[FL] Armazenamento em cache hierárquico com três proxies

Definir o tempo durante o qual as páginas devem ficar no cache é um pouco mais complicado. Algumas páginas não devem ser armazenadas no cache de modo algum. Por exemplo, uma página contendo os preços das 50 ações mais ativas muda a cada segundo. Se ela fosse armazenada no cache, um usuário que obtivesse uma cópia do cache receberia dados **ultrapassados** (isto é, obsoletos). Por outro lado, uma vez que a bolsa de valores tenha fechado o movimento do dia, essa página permanecerá válida durante horas ou dias, até começar a próxima sessão da bolsa. Desse modo, a utilidade de armazenar uma página no cache pode variar muito com o tempo.

A questão fundamental na determinação de quando evitar uma página proveniente do cache é saber até que ponto os usuários estão dispostos a aceitar dados ultrapassados (pois as páginas guardadas no cache são mantidas no disco, e assim o espaço de armazenamento consumido poucas vezes é um problema). Se um proxy descartar páginas com rapidez, raramente ele retornará uma página ultrapassada, mas também não será muito eficiente (isto é, terá uma taxa de acertos baixa). Se mantiver as páginas por um tempo muito longo, ele poderá ter uma alta taxa de acertos, mas em compensação com freqüência retornará páginas ultrapassadas.

Há duas abordagens para lidar com esse problema. A primeira utiliza uma heurística com a finalidade de avaliar por quanto tempo cada página deve ser mantida. Uma estratégia comum é basear o tempo de armazenamento no cabeçalho *Last-Modified* (consulte a Figura 7.43). Se uma página foi modificada há uma hora, ela será guardada no cache por uma hora. Se ela foi modificada há um ano, evidentemente é uma página muito estável (por exemplo, uma lista dos deuses da mitologia greco-romana) e assim pode ser guardada no cache por um ano com uma expectativa razoável de não se alterar durante esse período. Embora essa heurística quase sempre funcione bem na prática, de vez em quando ela retorna páginas ultrapassadas.

A outra abordagem é mais dispendiosa, mas elimina a possibilidade de páginas ultrapassadas usando recursos especiais da RFC 2616 que lidam com o gerenciamento de caches. Uma dos recursos mais úteis é o cabeçalho de solicitação *If-Modified-Since*, que um proxy pode enviar a um servidor. Ele especifica a página que o proxy deseja e a hora em que a página guardada no cache foi modificada pela última vez (do cabeçalho *Last-Modified*). Se a página não tiver sido modificada desde então, o servidor devolverá uma breve mensagem *Not Modified* (código de status 304 na Figura 7.42), que instrui o proxy a usar a página guardada no cache. Se a página tiver sido modificada desde então, a nova página será retornada. Embora essa abordagem sempre exija uma mensagem de solicitação e uma mensagem de resposta, a mensagem de resposta será muito curta quando a entrada do cache ainda for válida.

Essas duas abordagens podem ser facilmente combinadas. Durante o primeiro intervalo ΔT depois de buscar a página, o proxy simplesmente a devolve aos clientes que a solicitaram. Depois que a página permanece por algum tempo no cache, o proxy utiliza mensagens *If-Modified-Since* para conferir sua atualidade. A escolha de ΔT invariavelmente envolve algum tipo de heurística, dependendo

de quanto tempo faz que página foi modificada pela última vez.

As páginas da Web que guardam conteúdo dinâmico (por exemplo, geradas por um script de PHP) nunca devem ser guardadas no cache, pois os parâmetros podem ser diferentes da próxima vez. Para lidar com esse e com outros casos, existe um mecanismo geral que permite a um servidor instruir todos os proxies ao longo o caminho até o cliente para não usarem a página atual outra vez sem verificar sua atualidade. Esse mecanismo também pode ser usado no caso de qualquer página que possa mudar rapidamente. Vários outros mecanismos de controle de cache também são definidos na RFC 2616.

Outra abordagem para melhorar o desempenho é o armazenamento em cache proativo. Quando um proxy busca uma página em um servidor, ele pode inspecionar a página para ver se ela tem algum hiperlink. Nesse caso, o proxy pode emitir solicitações para os servidores relevantes, a fim de carregar previamente o cache com as páginas apontadas pelos links, caso elas sejam necessárias. Essa técnica pode reduzir o tempo de acesso em solicitações subseqüentes, mas também pode inundar as linhas de comunicações com páginas que nunca serão necessárias.

É claro que o armazenamento em caches da Web está longe de ser um assunto trivial. Poderíamos dizer muito mais sobre ele. Na verdade, foram escritos livros inteiros sobre esse tema, como (Rabinovich e Spatscheck, 2002; e Wessels, 2001). Porém, é hora de passarmos para o próximo tópico.

[T4] Replicação de servidores

O armazenamento em cache é uma técnica do lado cliente para melhorar o desempenho, mas também existem técnicas do lado servidor. A abordagem mais comum que os servidores adotam para melhorar o desempenho é replicar seu conteúdo em vários locais bem separados. Às vezes, essa técnica é chamada

espelhamento.

Em geral, o espelhamento é usado na principal página da Web de uma empresa, para guardar algumas imagens juntamente com links para, digamos, os Web sites da regiões leste, oeste, norte e sul da empresa. O usuário deve clicar no link do site mais próximo para acessar esse servidor. Daí em diante, todas as solicitações vão para o servidor selecionado.

Normalmente, os sites espelhados são estáticos. A empresa decide onde deseja colocar os espelhos, instala um servidor em cada região e coloca nesse servidor quase todo o conteúdo (talvez omitindo os limpadores de neve no site de Miami e as cadeiras de praia no site do Alasca). De um modo geral, a escolha dos locais permanece estável durante meses ou anos.

Infelizmente, a Web tem um fenômeno conhecido como **sucesso instantâneo**, no qual um Web site que antes era desconhecido e pouco visitado, de repente se torna o centro do universo virtual. Por exemplo, até 6 de novembro de 2000, o Web site da secretaria de estado da Flórida, *www.dos.state.fl.us*, oferecia on-line alguns minutos das reuniões realizadas no gabinete do estado da Flórida e instruções como se tornar um tabelião na Flórida. No entanto, em 7 de novembro de 2000, quando a presidência dos EUA repentinamente passou a depender de alguns milhares de votos disputados em meia dúzia de condados da Flórida, ele passou a ser um dos cinco Web sites mais visitados do mundo. É desnecessário dizer que ele não conseguiu manipular a carga e quase morreu tentando.

É necessário encontrar um meio para que um Web site, ao perceber um aumento maciço e repentino em seu tráfego, possa criar clones de si mesmo automaticamente em muitos locais de acordo com a necessidade, manter esses sites em operação até a tempestade passar, e depois desativar muitos deles ou até mesmo todos. Para ter essa habilidade, um site precisa estabelecer um acordo com alguma empresa que possua muitos sites de hospedagem e que possa criar

réplicas conforme a demanda, desde que o site pague pela capacidade que realmente utilizar.

Uma estratégia ainda mais flexível é criar réplicas dinâmicas de cada página, dependendo da origem do tráfego. Algumas pesquisas sobre esse tópico são relatadas em (Pierre *et al.*, 2001; e Pierre *et al.*, 2002).

[T4] Redes de entrega de conteúdo

Uma das maravilhas do capitalismo é a possibilidade de ganhar dinheiro na World Wide Wait. Vamos ver como esse mecanismo funciona. Empresas denominadas **CDNs (Content Delivery Networks)** se comunicam com provedores de conteúdo (sites de música, de jornais e outros que queiram tornar seu conteúdo disponível de modo rápido e fácil) e se oferecem para entregar esse conteúdo aos usuários de modo eficiente, em troca de uma taxa. Depois que o contrato é assinado, o proprietário do conteúdo fornece à CDN o conteúdo do seu Web site para pré-processamento (o que discutiremos em breve) e posterior distribuição.

Em seguida, a CDN se comunica com grandes números de ISPs e oferece pagamento pela permissão para colocar em suas LANs um volumoso servidor gerenciado remotamente com um conteúdo valioso. Isso não apenas é uma fonte de renda, mas também oferece aos clientes do ISP um excelente tempo de resposta nos acessos ao conteúdo da CDN, dando assim ao ISP uma vantagem competitiva sobre outros ISPs que não aceitaram o dinheiro oferecido pela CDN. Sob essas condições, a assinatura de um contrato com uma CDN não traz nenhuma despesa para o ISP. Como conseqüência, as maiores CDNs têm mais de 10.000 servidores distribuídos por todo o mundo.

Com o conteúdo replicado em milhares de sites por todo o mundo, é claro que existe um grande potencial para melhorar o desempenho. Entretanto, para que tudo funcione, tem de haver um meio de redirecionar a solicitação do cliente para

o servidor da CDN mais próximo, de preferência um que esteja colocado no ISP do cliente. Além disso, esse redirecionamento deve ser feito sem modificar o DNS ou qualquer outra parte da infra-estrutura padrão da Internet. Apresentamos a seguir uma descrição ligeiramente simplificada de como a Akamai, a maior CDN, consegue isso.

O processo inteiro começa quando o provedor de conteúdo entrega à CDN seu Web site. Então, a CDN faz cada página passar por um pré-processador que substitui todos os URLs por outros modificados. No modelo funcional que rege essa estratégia, o Web site do provedor de conteúdo consiste em muitas páginas minúsculas (apenas texto em HTML), mas essas páginas muitas vezes têm links para grandes arquivos, como imagens, áudio e vídeo. As páginas de HTML modificadas são armazenadas no servidor do provedor de conteúdo e são acessadas da maneira habitual; as imagens, o áudio e o vídeo vão para os servidores da CDN.

Para ver como esse esquema realmente funciona, considere a página da Web da Furry Video da Figura 7.46(a). Depois do pré-processamento, ela é transformada, ficando com a aparência da Figura 7.46(b), e é colocada no servidor da Furry Video como *www.furryvideo.com/index.html*.

Quando um usuário digita o URL *www.furryvideo.com*, o DNS retorna o endereço IP do Web site da Furry Video, permitindo que a página principal (HTML) seja acessada da maneira normal. Quando há um clique em qualquer dos hiperlinks, o navegador pede ao DNS para procurar *cdn-server.com*. O navegador então envia uma solicitação HTTP para esse endereço IP, esperando receber um arquivo MPEG.

Isso não acontece, porque *cdn-server.com* não hospeda qualquer conteúdo. Em vez disso, ele é o falso servidor HTTP da CDN. O servidor examina o nome do arquivo e o nome do servidor para descobrir qual é a página necessária e de qual

provedor de conteúdo. Ele também examina o endereço IP da solicitação recebida e pesquisa em seu banco de dados para determinar a provável origem do usuário. De posse dessas informações, ele determina qual dos servidores de conteúdo da CDN pode oferecer o melhor serviço ao usuário. Essa decisão é difícil, porque o servidor mais próximo geograficamente pode não ser o mais próximo em termos de topologia de rede, e o mais próximo em termos de topologia de rede pode estar muito ocupado no momento. Depois de fazer uma escolha, *cdn-server.com* envia uma resposta com o código de status 301 e um cabeçalho *Location* fornecendo o URL do arquivo no servidor de conteúdo da CDN mais próximo do cliente. Para esse exemplo, vamos supor que o URL é *www.CDN-0420.com/furryvideo/bears.mpg*. Em seguida, o navegador processa esse URL da maneira habitual para obter o arquivo MPEG real.

As etapas envolvidas são ilustradas na Figura 7.47. A primeira etapa é pesquisar *www.furryvideo.com* para obter seu endereço IP. Depois disso, a página de HTML pode ser buscada e exibida da maneira usual. A página contém três hiperlinks para *cdn-server* [veja a Figura 7.46(b)]. Quando, digamos, o primeiro hiperlink é selecionado, seu endereço DNS é pesquisado (etapa 5) e retornado (etapa 6). Quando uma solicitação para *bears.mpg* é enviada para *cdn-server* (etapa 7), o cliente é informado de que deve ir para *CDN-0420.com* (etapa 8). Ao seguir a instrução (etapa 9), ele recebe o arquivo do cache do proxy (etapa 10). A propriedade que faz todo esse mecanismo funcionar é a etapa 8, o falso servidor HTTP que redireciona o cliente para um proxy da CDN próximo ao cliente.

[arte: ver original da p. 662]

[TD]

```
<html>
<head> <title> Furry Video </title> </head>
<body>
```

```
<h1> Furry Video's Product List </h1>

<p> Click below for free samples. </p>


<a href="bears.mpg"> Bears Today </a> <br>

<a href="bunnies.mpg"> Funny Bunnies </a> <br>

<a href="mice.mpg"> Nice Mice </a> <br>

</body>

</html> [TN]
```

(a)

[TD]

```
<html>

<head> <title> Furry Video </title> </head>

<body>

<h1> Furry Video's Product List </h1>

<p> Click below for free samples. </p>


<a href="http://cdn-server.com/furryvideo/bears.mpg"> Bears Today </a>

<br>

<a href="http://cdn-server.com/furryvideo/bunnies. mpg"> Funny Bunnies </a>

<br>

<a href="http://cdn-server.com/furryvideo/mice.mpg"> Nice Mice </a> <br>

</body>

</html> [TN]
```

(b)

[F]Figura 7.46

[FL] (a) Página da Web original. (b) A mesma página depois de ser transformada

Normalmente, o servidor da CDN para o qual o cliente é redirecionado é um proxy com um grande cache pré-carregado com o conteúdo mais importante. Porém, se alguém solicitar um arquivo que não está no cache, ele será buscado no servidor verdadeiro e inserido no cache para uso subseqüente. Fazendo do servidor de conteúdo um proxy em vez de uma réplica completa, a CDN tem habilidade para negociar o tamanho do disco, o tempo de pré-carregamento e os vários parâmetros de desempenho.

Para obter mais informações sobre as redes de entrega de conteúdo, consulte (Hull, 2002; e Rabinovich e Spatscheck, 2002).

[T3] 7.3.6 A Web sem fios

Há um interesse considerável em pequenos dispositivos portáteis capazes de acessar a Web por meio de um link sem fio. De fato, as primeiras tentativas nesse sentido já foram feitas. Sem dúvida haverá uma grande mudança nessa área nos próximos anos, mas ainda vale a pena examinar algumas idéias atuais relacionadas à Web sem fio, para ver onde estamos agora e para onde podemos seguir. Vamos nos concentrar nos dois primeiros sistemas geograficamente distribuídos da Web sem fio que chegaram ao mercado: WAP e i-mode.

[arte: ver original p. 663]

[Dísticos]

[1] Servidor DNS

1 2 5 6

[2] www.furryvideo.com 3 Cliente 7 cdn-server (HTTP falso)

4 8

[3] 9 10

Cache em disco

Proxy

CDN-0420.com

[4]

1. Pesquisar www.furryvideo.com

2. Endereço IP da Furry retornado

3. Página HTML da Furry solicitada

4. Página HTML retornada

5. Depois do clique, pesquisar cdn-server.com

6. Endereço IP de cdn-server retornado

7. Solicitar bears.mpg a cdn-server

8. Cliente informado de que deve se redirecionar para CDN-0420.com

9. Solicitação de bears.mpg

10. Arquivo do cache bears.mpg retornado

[F]Figura 7.47

[FL] Etapas na pesquisa de um URL quando é usada uma CDN

[T4] WAP — Wireless Application Protocol

Depois que a Internet e os telefones celulares se tornaram comuns, não demorou muito para que alguém tivesse a idéia de combiná-los em um telefone móvel com uma tela embutida para acesso sem fio ao correio eletrônico e à Web. Nesse caso, "alguém" foi um consórcio liderado inicialmente pelas empresas Nokia, Ericsson, Motorola e phone.com (antes denominada Unwired Planet) e que tem agora centenas de membros. O sistema é chamado **WAP (Wireless Application Protocol).** Um dispositivo WAP pode ser um telefone móvel aperfeiçoado, um PDA ou um notebook sem qualquer recurso de voz. A especificação permite tudo isso e muito mais. A idéia básica é usar a infra-estrutura sem fio digital existente. Os usuários podem literalmente chamar um gateway WAP pelo link sem fio e enviar solicitações de página da Web. Em seguida, o gateway verifica se seu cache tem a

página solicitada. Se estiver presente, ela será enviada; caso contrário, o gateway irá buscá-la na Internet fisicamente conectada. Em essência, isso significa que o WAP 1.0 é um sistema de comutação de circuitos com uma tarifa de conexão por minuto bastante alta. Para encurtar a história, as pessoas não gostaram de acessar a Internet usando uma tela minúscula e pagando por minuto; assim, o WAP se tornou um fiasco (embora também houvesse outros problemas). Porém, o WAP e seu concorrente, o i-mode (descrito a seguir) parecem estar convergindo para uma tecnologia semelhante e, desse modo, o WAP 2.0 ainda poderá ser um grande sucesso. Tendo em vista que o WAP 1.0 foi a primeira tentativa de utilização da Internet sem fio, vale a pena descrevê-lo, pelo menos brevemente.

Em essência, o WAP é uma pilha de protocolos para acesso à Web, embora otimizada para conexões de baixa largura de banda, usando dispositivos sem fios com uma CPU lenta, pouca memória e uma tela pequena. Esses requisitos são evidentemente distintos das exigências que os PCs de desktop padrão devem manter, o que resulta em algumas diferenças nos protocolos. As camadas são mostradas na Figura 7.48.

[arte: ver original p. 664]

[Dísticos]

[1] Wireless application environment (WAE)

[2] Wireless session protocol (WSP)

[3] Wireless transaction protocol (WTP)

[4] Wireless transport layer security (WTLS)

[5] Wireless datagram protocol (WDP)

[6] Camada portadora (GSM, CDMA, D-AMPS, GPRS etc.)

[F]Figura 7.48

[FL] A pilha de protocolos do WAP

A camada mais baixa admite todos os sistemas de telefonia móvel existentes, inclusive GSM, D-AMPS e CDMA. A taxa de dados do WAP 1.0 é 9600 bps. Sobre essa camada encontra-se o protocolo de datagramas, o **WDP (Wireless Datagram Protocol)**, que é essencialmente o UDP. Em seguida, há uma camada para segurança, sem dúvida necessária em um sistema sem fios. O WTLS é um subconjunto da SSL da Netscape, que estudaremos no Capítulo 8. Acima dessa camada, temos uma camada de transações, que administra solicitações e respostas, tanto de forma confiável quanto não confiável. Essa camada substitui o TCP, que não é utilizado no enlace aéreo por razões de eficiência. Depois, existe uma camada de sessão, semelhante ao HTTP/1.1, mas com algumas restrições de extensões para fins de otimização. A camada superior é um micronavegador (WAE).

Além do custo, o outro aspecto que sem dúvida prejudicou a aceitação do WAP foi o fato de que ele não utiliza a HTML. Em vez disso, a camada WAE usa uma linguagem de marcação chamada **WML (Wireless Markup Language)**, uma aplicação de XML. Como conseqüência, em princípio, um dispositivo WAP só pode acessar as páginas convertidas para WML. Porém, tendo em vista que isso restringe bastante o valor do WAP, a arquitetura exige um filtro em tempo de execução de HTML para WML, a fim de aumentar o conjunto de páginas disponíveis. Essa arquitetura é ilustrada na Figura 7.49.

Com toda justiça, provavelmente o WAP estava à frente de seu tempo. Quando o WAP foi lançado, a XML não era conhecida fora do W3C, e assim a imprensa relatou seu lançamento como **O WAP NÃO USA HTML.** Uma manchete mais precisa seria: **O WAP JÁ UTILIZA O NOVO PADRÃO HTML.** Porém, depois de tudo isso, ficou difícil reparar os danos, e WAP 1.0 nunca mais se firmou. Estudaremos novamente o WAP depois de examinarmos seu principal concorrente.

[arte: ver original p. 665]

[Dísticos]

[1] Estação base

[2] Filtro de HTML para WML

WML HTML

WML

[3] Servidor da Web

Internet

[4] Gateway WAP

[5] WTP

Dispositivo WAP

[F]Figura 7.49

[FL] A arquitetura WAP

[T4] I-Mode

Enquanto um consórcio de várias empresas reunindo fornecedores de produtos telecomunicações e empresas de informática estava ocupado na elaboração de um padrão aberto que utilizasse a versão de HTML mais avançada disponível, outros desenvolvimentos estavam ocorrendo no Japão. Lá, uma mulher japonesa, Mari Matsunaga, desenvolveu uma abordagem diferente para a Web sem fio, chamada **i-mode (information-mode).** Ela convenceu a subsidiária de comunicações sem fios do antigo monopólio de telefonia japonês que sua abordagem era correta e, em fevereiro de 1999, a NTT DoCoMo (literalmente: Japanese Telephone and Telegraph Company — onde quer que você vá) lançou o serviço no Japão. Em 3 anos, ela tinha mais de 35 milhões de assinantes japoneses, que podiam acessar mais de 40.000 Web sites especiais de i-mode. A empresa também fez a maior parte das empresas de telecomunicações do mundo se entusiasmarem com seu sucesso financeiro, especialmente considerando-se o

fato de que o WAP parecia não estar indo a lugar nenhum. Agora, vamos examinar o que é o i-mode e como ele funciona.

O sistema i-mode tem três componentes importantes: um novo sistema de transmissão, um novo aparelho e uma nova linguagem para projeto de página da Web. O sistema de transmissão consiste em duas redes separadas: a rede de telefonia móvel comutada por circuitos existente (de certa forma comparável ao D-AMPS) e uma nova rede comutada por pacotes construída especificamente para o serviço do i-mode. O modo de voz utiliza a rede comutada por circuitos e é tarifado por minuto de tempo de conexão. O i-mode utiliza a rede comutada por pacotes e está sempre ativa (como ADSL ou cabo), e assim não há nenhuma cobrança pelo tempo de conexão. Em vez disso, há uma tarifa por cada pacote enviado. Atualmente, não é possível usar ambas as redes ao mesmo tempo.

Os aparelhos são semelhantes aos telefones celulares, com a inclusão de uma pequena tela. A NTT DoCoMo anuncia os dispositivos i-mode como telefones celulares aperfeiçoados e não como terminais da Web sem fios, embora eles sejam exatamente isso. De fato, é provável que a maioria dos clientes não esteja sequer consciente de que está na Internet. Eles pensam que seus dispositivos i-mode são telefones celulares com serviços aperfeiçoados. Mantendo a idéia de que esse modelo de i-mode é um serviço, os aparelhos não são programáveis pelo usuário, embora contenham o equivalente a um PC de 1995 e provavelmente possam executar o Windows 95 ou o UNIX.

Quando o aparelho i-mode é ligado, o usuário fica diante de uma lista de categorias dos serviços aprovados oficialmente. Existem bem mais de 1000 serviços divididos em cerca de 20 categorias. Cada serviço, que na realidade é um pequeno Web site do i-mode, é administrado por uma empresa independente. As principais categorias no menu oficial incluem correio eletrônico, notícias, previsão do tempo, esportes, jogos, compras, mapas, horóscopos,

entretenimento, viagens, guias regionais, tons de discagem, receitas, apostas, home banking e cotações de ações. De certa forma, o serviço é destinado aos adolescentes e às pessoas na faixa dos 20 anos, que tendem a adorar pequenos aparelhos eletrônicos, em especial se eles tiverem cores vistosas. O mero fato de que mais de 40 empresas estão vendendo tons de discagem já significa algo. A aplicação mais popular é o correio eletrônico, que permite mensagens de até 500 bytes, e portanto é visto como um grande avanço em relação ao SMS (Short Message Service) com suas mensagens de 160 bytes. Os jogos também são populares.

Existem também mais de 40.000 Web sites do i-mode, mas eles têm de ser acessados digitando-se seus URLs, em vez de selecioná-los a partir de um menu. De certo modo, a lista oficial é como um portal da Internet que permite que outros Web sites sejam acessados com um clique, em vez de se digitar um URL.

A NTT DoCoMo controla com firmeza os serviços oficiais. Para ser aceito na lista, um serviço deve atender a uma grande variedade de critérios publicados. Por exemplo, um serviço não deve ter uma influência ruim sobre a sociedade, os dicionários japonês-inglês devem ter palavras suficientes, os serviços com tons de discagem têm de acrescentar novos tons com freqüência, e nenhum site pode criticar o comportamento ou a atuação da NTT DoCoMo (Frengle, 2002). Os 40.000 sites da Internet podem fazer o que quiserem.

O modelo comercial do i-mode é tão diferente do modelo da Internet convencional que vale a pena explicá-lo. A taxa básica de assinatura do i-mode é de alguns dólares por mês. Tendo em vista que existe uma tarifa para cada pacote recebido, a assinatura básica inclui um pequeno número de pacotes. Como alternativa, o cliente pode escolher uma assinatura com mais pacotes gratuitos, com a tarifa por pacote caindo nitidamente à medida que você passa de 1 MB por mês para 10 MB por mês. Se os pacotes gratuitos forem usados antes

de completar o mês, é possível adquirir pacotes adicionais on-line.

Para usar um serviço, você tem de assiná-lo; para isso, basta clicar sobre ele na lista e digitar seu código PIN. A maioria dos serviços oficiais custa em torno de 1 a 2 dólares por mês. A NTT DoCoMo inclui a tarifa na conta telefônica e repassa 91% dela para o provedor de serviços, ficando com 9%. Se um serviço não oficial tiver 1 milhão de clientes, ele terá de enviar 1 milhão de contas cobrando (aproximadamente) 1 dólar a cada mês. Se esse serviço se tornar oficial, a NTT DoCoMo cuidará da cobrança e simplesmente irá transferir 910.000 dólares todo mês para a conta bancária do serviço. Não ter de cuidar da cobrança é um enorme incentivo para um provedor de serviços se tornar oficial, o que gera mais lucro para a NTT DoCoMo. Além disso, ser oficial significa incluir sua empresa no menu inicial, o que torna seu site muito mais fácil de ser encontrado. A conta telefônica do usuário inclui ligações telefônicas, tarifas de assinatura do i-mode, tarifas de assinatura de serviços e pacotes extras.

Apesar de seu estrondoso sucesso no Japão, não está claro se ele terá sucesso nos Estados Unidos e na Europa. Em certos aspectos, a situação japonesa é diferentes da situação nos países do Ocidente. Primeiro, a maioria dos clientes potenciais no Ocidente (por exemplo, adolescentes, alunos universitários e pessoas da área de negócios) já tem um PC de tela grande em casa, muito provavelmente com uma conexão para a Internet a uma velocidade de pelo menos 56 kbps, em geral muito maior. No Japão, poucas pessoas têm um PC conectado à Internet em casa, em parte devido à falta de espaço, mas também devido às tarifas exorbitantes da NTT pelos serviços de telefonia local (algo como 700 dólares pela instalação de uma linha e 1,50 dólar por hora para chamadas locais). Para a maioria dos usuários, o i-mode é a única conexão da Internet.

Em segundo lugar, as pessoas que vivem no Ocidente não estão habituadas a pagar 1 dólar por mês para acessar o Web site da CNN, 1 dólar por mês para

acessar o Web site do Yahoo, 1 dólar por mês para acessar o Web site do Google e assim por diante, sem mencionar alguns dólares por MB transferido por download. A maioria dos provedores da Internet do Ocidente cobra uma tarifa fixa mensal independente da utilização real, em grande parte como uma resposta à demanda dos clientes.

Em terceiro lugar, para muitos japoneses, o principal horário de uso do i-mode ocorre enquanto eles estão indo ou vindo do trabalho ou da escola no trem ou no metrô. Na Europa, bem menos pessoas vão trabalhar de trem e, nos Estados Unidos, quase ninguém faz isso. Usar o i-mode em casa perto de um computador com um monitor de 17 polegadas, uma conexão ADSL de 1 Mbps e todos os megabytes gratuitos que desejar não faz muito sentido fora do Japão. Apesar disso, ninguém previa a imensa popularidade dos telefones celulares, e portanto o i-mode ainda pode encontrar um nicho de mercado no Ocidente.

Como mencionamos antes, os aparelhos do i-mode utilizam a rede comutada por circuitos existente para voz e uma nova rede comutada por pacotes para transmissão de dados. A rede de dados se baseia no CDMA e transmite pacotes de 128 bytes a 9600 bps. Um diagrama da rede é apresentado na Figura 7.50. Os aparelhos se comunicam usando **LTP (Lightweight Transport Protocol)** sobre o link aéreo com um gateway de conversão de protocolos. O gateway tem uma conexão de banda larga de fibra óptica para o servidor i-mode, que está conectado a todos os serviços. Quando o usuário seleciona um serviço no menu oficial, a solicitação é enviada ao servidor i-mode, que armazena no cache a maioria das páginas, a fim de melhorar o desempenho. As solicitações a sites que não estão no menu oficial ignoram o servidor i-mode e passam diretamente pela Internet.

Os aparelhos atuais têm CPUs que funcionam em aproximadamente 100 MHz, vários megabytes de ROM flash, talvez 1 MB de RAM e uma pequena tela

embutida. O i-mode exige que a tela tenha pelo menos 72 × 94 pixels, mas alguns dispositivos de ponta têm até 120 × 160 pixels. Em geral, as telas têm cores de 8 bits, o que permite 256 cores. Isso não é suficiente para fotografias, mas é adequado para desenhos a traço e caricaturas simples. Tendo em vista que não existe nenhum mouse, a navegação na tela é feita com as teclas de setas.

A estrutura de software é mostrada na Figura 7.51. A camada inferior consiste em um sistema operacional simples de tempo real para controlar o hardware. Em seguida, temos um módulo que realiza a comunicação de rede, usando o protocolo LTP patenteado da NTT DoCoMo. Acima dele, há um gerenciador de janelas simples que manipula o texto e gráficos simples (arquivos GIF). Com telas de apenas 120 × 160 pixels no máximo, não há muito para administrar.

[arte: ver original p. 668a]

[Dísticos]

[1] Estação base

LTP

Aparelho i-mode

[2] LTP

Para rede de voz

Gateway de conversão de protocolos

[3] Servidor i-mode

TCP

TCP

Conexão direta para a Internet

[4] Linha dedicada Provedor de serviços Serviços do menu oficial

Internet

[F]Figura 7.50

[FL] A estrutura da rede de dados do i-mode, mostrando os protocolos de

[arte: ver original p. 668b]

[Dísticos]

[1] Módulo de interação de usuário

[2] Plug-ins Interpretador cHTML Java

[3] Gerenciador de janelas simples

[4] Comunicação de rede

[5] Sistema operacional de tempo real

[F]Figura 7.51

[FL] Estrutura do software do i-mode

A quarta camada contém o interpretador de páginas da Web (isto é, o navegador). O i-mode não utiliza a HTML completa, mas um subconjunto dela, chamado **cHTML (compact HTML)**, baseado livremente na HTML 1.0. Essa camada também permite aplicações auxiliares e plug-ins, da mesma forma que os navegadores dos PCs. Uma aplicação auxiliar padrão é um interpretador para uma versão ligeiramente modificada da JVM.

No nível superior, há um módulo de interação do usuário, que administra a comunicação com o usuário.

Agora vamos observar com mais detalhes a cHTML. Como mencionamos, ela é semelhante à HTML 1.0, com algumas omissões e algumas extensões para uso em aparelhos móveis. Ela foi submetida ao W3C para padronização, mas o W3C mostrou pouco interesse na linguagem, e assim é provável que ela continue a ser um produto patenteado.

A maior parte das tags básicas de HTML é permitida, incluindo [TD]<html>[TN], [TD]<head>[TN], [TD]<title>[TN], [TD]<body>[TN], [TD]<hn>[TN],

[TD]<center>[TN], [TD]<ul>[TN], [TD]<ol>[TN], [TD]<menu>[TN], [TD]<li>[TN], [TD]
[TN], [TD]<p>[TN], [TD]<hr>[TN], [TD]<img>[TN], [TD]<form>[TN] e [TD]<input>[TN]. As tags [TD]<b>[TN] e [TD]<i>[TN] não são permitidas.

A tag [TD]<a>[TN] é permitida para vinculação a outras páginas, mas com o esquema adicional *tel* para números de telefones de discagem de. De certo modo, *tel* é análogo a *mailto*. Quando um hiperlink que utiliza o esquema *mailto* é selecionado, o navegador exibe um formulário para envio de correio eletrônico ao destino identificado no link. Quando um hiperlink que utiliza o esquema *tel* é selecionado, o navegador disca o número de telefone. Por exemplo, um catálogo de endereços poderia ter imagens simples de várias pessoas. Ao selecionar uma delas, o aparelho ligaria para a pessoa. A RFC 2806 discute os URLs de telefone.

O navegador cHTML é limitado em outros aspectos. Ele não admite JavaScript, quadros, folhas de estilos, cores de fundo ou imagens de fundo. Ele também não admite imagens JPEG, porque elas demoram tempo demais até serem descompactadas. Os miniaplicativos Java são permitidos mas (atualmente) são limitados a 10 KB, devido à baixa velocidade de transmissão no link aéreo.

Embora a NTT DoCoMo tenha removido algumas tags de HTML, ela também adicionou outras tags novas. A tag [TD]<blink>[TN] faz o texto aparecer e desaparecer (piscar). Embora possa parecer inconsistente proibir [TD]<b>[TN] (na suposição de que os Web sites não devem lidar com a aparência) e depois acrescentar [TD]<blink>[TN] que se relaciona apenas à aparência, foi isso que a empresa fez. Outra tag nova é [TD]<marquee>[TN], que efetua a rolagem de seu conteúdo na tela, como um marcador de cotação de ações.

Uma nova característica é o atributo *align* para a tag [TD]
[TN]. Ele é necessário porque, com uma tela de 6 linhas de 16 caracteres, há um grande perigo de certas palavras serem quebradas, como mostra a Figura 7.52(a). O atributo *Align* ajuda a reduzir esse problema, tornando possível obter algo mais

parecido com a Figura 7.52(b). É interessante notar que o idioma japonês não têm o problema de quebras de palavras em linhas. Para um texto kanji, a tela é dividida em uma grade retangular de células com o tamanho 9 × 10 pixels ou 12 × 12 pixels, dependendo da fonte usada. Cada célula contém exatamente um caracteres kanji, o equivalente a uma palavra em Inglês. Quebras de linhas entre palavras são sempre permitidas em japonês.

[arte: ver original p. 669]

[Dísticos]

[1]

[TD]The time has com

e the walrus sai

d to talk of man

y things. Of sho

es and ships and

sealing wax of c[TN]

(a)

[2]

[TD]The time has

come the walrus

said to talk of

many things. Of

shoes and ships

and sealing wax [TN]

(b)

[F]Figura 7.52

[FL] Lewis Carroll encontra a tela de 16 × 6

Embora o idioma japonês tenha dezenas de milhares de kanji, a NTT DoCoMo criou 166 inteiramente novos, chamados **emoji**, com um fator de interesse mais alto — são essencialmente pictogramas semelhantes aos smileys da Figura 7.6. Eles incluem símbolos para os signos astrológicos, cerveja, hambúrguer, parque de diversões, aniversário, telefone celular, cachorro, gato, Natal, coração partido, beijo, humor, soneca e, é claro, um que significa beleza.

Outro atributo novo é a habilidade para permitir que os usuários selecionem hiperlinks usando o teclado, sem dúvida uma propriedade importante em um computador sem mouse. Um exemplo de utilização desse atributo é usado no arquivo cHTML da Figura 7.53.

[arte: ver original da p. 670]

[TD]

```
<html>
<body>
<h1> Select an option </h1>
<a href="messages.chtml" accesskey="1"> Check voicemail </a> <br>
<a href="mail.chtml" accesskey="2"> Check e-mail </a> <br>
<a href="games.chtml" accesskey="3"> Play a game </a>
</body>
</html> [TN]
```

[F]Figura 7.53

[FL] Um exemplo de arquivo cHTML

Embora o lado cliente seja um pouco limitado, o servidor i-mode é um computador repleto de recursos, com todos os penduricalhos habituais. Ele admite CGI, Perl, PHP, JSP, ASP e tudo que os outros servidores da Web normalmente admitem.

Uma comparação breve entre o WAP e o i-mode que foi realmente implementado nos sistemas de primeira geração é apresentada na Figura 7.54. Embora algumas diferenças possam parecer pequenas, freqüentemente elas são importantes. Por exemplo, os jovens de 15 anos não tem cartões de crédito; assim, a possibilidade de fazer comprar por e-commerce e debitá-las na conta telefônica faz uma grande diferença no momento de optar por sistema.

Para obter informações adicionais sobre i-mode, consulte (Frengle, 2002; e Vacca, 2002).

[T4] Web sem fio da segunda geração

O WAP 1.0, baseado em padrões internacionais reconhecidos, deveria ser uma ferramenta séria para as pessoas da área de negócios enquanto estão em trânsito, mas esse sistema foi malsucedido. O i-mode era um brinquedo eletrônico para os adolescentes japoneses, utilizando tudo em formato patenteado, e foi um enorme sucesso. O que aconteceu? Cada grupo aprendeu algo com a primeira geração da Web sem fio. O consórcio do WAP aprendeu que o conteúdo é importante. Não ter um grande número de Web sites que se comuniquem usando sua linguagem de marcação é fatal. A NTT DoCoMo aprendeu que um sistema fechado e patenteado intimamente relacionado a minúsculos aparelhos e à cultura japonesa não é um bom produto de exportação. Os dois lados concluíram que, para convencer um grande número de Web sites a colocar seu conteúdo em seu formato patenteado, é necessário ter uma linguagem de marcação aberta e estável, de aceitação universal. Guerras de formatos não são boas para os negócios.

Ambos os serviços estão prestes a entrar na segunda geração da tecnologia da Web sem fio. O WAP 2.0 chegou primeiro, e assim vamos usá-lo como nosso exemplo. O WAP 1.0 fez algumas coisas certas, e teve continuidade. Por um lado,

o WAP pode ser usado em uma variedade de redes diferentes. A primeira geração empregava redes de comutação de circuitos, mas as redes de comutação de pacotes sempre foram e ainda são uma opção. Os sistemas de segunda geração deverão usar a comutação de pacotes como, por exemplo, o GPRS. Por outro lado, o WAP se destinava inicialmente a dar suporte a uma ampla variedade de dispositivos, desde telefones celulares até poderosos notebook, e ainda tem esse objetivo.

[arte: ver original p. 671]

[T]Tabela

Característica

O que ele é

Dispositivo

Acesso

Rede subjacente

Taxa de dados

Tela

Linguagem de marcação

Linguagem de scripts

Tarifas de utilização

Pagamento pelas compras

Pictogramas

Padronização

Onde é usado

Usuário típico

WAP

Pilha de protocolos

Aparelho, PDA, notebook

Discagem

Comutada por circuitos

9600 bps

Monocromática

WML (aplicação de XML)

WMLscript

Por minuto

Cartão de crédito

Não

Padrão aberto do fórum WAP

Europa, Japão

Homem de negócios

i-mode

Serviço

Aparelho

Sempre ativo

Duas: circuitos + pacotes

9600 bps

Em cores

cHTML

Nenhuma

Por pacote

conta telefônica

Sim

Patenteado pela NTT DoCoMo

Japão

Mulher jovem

[F]Figura 7.54

[FL] Uma comparação entre a primeira geração do WAP e a do i-mode

O WAP 2.0 também tem alguns recursos novos. Os mais significativos são:

1. Modelo de push, bem como modelo de pull.

2. Suporte para integração de telefonia a aplicações.

3. Sistema de mensagens de multimídia.

4. Inclusão de 264 pictogramas.

5. Interface para um dispositivo de armazenamento.

6. Suporte para plug-ins no navegador.

O modelo de pull é bem conhecido: o cliente solicita uma página e a recebe. O modelo de push admite dados que chegam sem serem solicitados, como um fluxo contínuo de cotações de ações ou de alertas de tráfego.

Voz e dados estão começando a se misturar, e o WAP 2.0 os admite em diversas formas. Já vimos um exemplo desse fato com a habilidade do i-mode para unir um ícone ou um texto na tela a um hiperlink com a forma de um número de telefone a ser chamado. Além de correio eletrônico e telefonia, também é admitido um sistema de mensagens de multimídia.

A enorme popularidade dos emojis do i-mode estimulou o consórcio do WAP a criar 264 emojis próprios. As categorias incluem animais, eletrodomésticos, vestuário, emoção, alimentos, corpo humano, sexo, mapas, música, plantas, esportes, hora, ferramentas, veículos, armas e tempo. É interessante observar que o padrão simplesmente denomina cada pictograma; ele não oferece o mapa de bits real, talvez com receio de que a representação de "soneca" ou "abraço" em uma cultura possa ser um insulto em outra cultura. O i-mode não teve esse problema, porque se destinava a um único país.

Fornecer uma interface de armazenamento não significa que todo telefone WAP

2.0 virá com um enorme disco rígido. A ROM flash também é um dispositivo de armazenamento. Uma câmera sem fio capaz de reconhecer o WAP poderia usar a ROM flash como espaço de armazenamento temporário de imagens, antes de transmitir as melhores imagens para a Internet.

Por fim, os plug-ins podem estender os recursos do navegador. Uma linguagem de scripts também é fornecida.

Diversas diferenças técnicas também estão presentes no WAP 2.0. As duas maiores se referem à pilha de protocolos e à linguagem de marcação. O WAP 2.0 continua a admitir a antiga pilha de protocolos da Figura 7.48, mas também admite a pilha da Internet padrão com TCP e HTTP/1.1. No entanto, foram feitas quatro mudanças secundárias (mas compatíveis) no TCP (para simplificar o código): (1) usar uma janela fixa de 64 KB, (2) nada de início lento, (3) uma MTU máxima de 1500 bytes e (4) um algoritmo de retransmissão um pouco diferente. O TLS é o protocolo de segurança da camada de transporte padronizado pela IETF; vamos examiná-lo no Capítulo 8. Muitos dispositivos iniciais provavelmente conterão ambas as pilhas, como mostra a Figura 7.55.

[arte: ver original p. 672]

[Dísticos]

[1] XHTML

[2] WSP HTTP

[3] WTP TLS

[4] WTLS TCP

[5] WDP IP

[6] Camada de portadora Camada de portadora

[7] Pilha de protocolos do WAP 1.0 Pilha de protocolos do WAP 2.0

[F]Figura 7.55

[FL] O WAP 2.0 admite duas pilhas de protocolos

A outra diferença técnica em relação ao WAP 1.0 é a linguagem de marcação. O WAP 2.0 admite XHTML Basic, destinada a pequenos dispositivos sem fios. Tendo em vista que a NTT DoCoMo também concordou em aceitar esse subconjunto, os projetistas de Web sites podem suar esse formato e ter certeza de que suas páginas funcionarão na Internet fixa e em todos os dispositivos sem fios. Essas decisões encerrarão as guerras de formatos de linguagem de marcação que estavam impedindo o crescimento do mercado da Web sem fio.

Algumas palavras sobre a XHTML Basic talvez sejam oportunas. Ela foi criada para telefones celulares, televisores, PDAs, máquinas automáticas, pagers, carros, máquinas de jogos e até mesmo relógios. Por essa razão, não admite folhas de estilos, scripts ou quadros, mas a maior parte das tags padrão está presente. Elas são agrupadas em 11 módulos. Alguns são obrigatórios e outros são opcionais, mas todos são definidos em XML. Os módulos e alguns exemplos de tags estão listados na Figura 7.56. Não examinamos detalhadamente todos os exemplos e tags, mas é possível obter mais informações em *www.w3.org*.

[arte: ver original p. 673]

[T]Tabela

Módulo

Structure

Text

Hypertext

List

Forms

Tables

Image

Object

Meta-information

Link

Base

Obrig.?

Sim

Sim

Sim

Sim

Não

Não

Não

Não

Não

Não

Não

Função

Estrutura de documentos

Informações

Hiperlinks

Listas de itens

Formulários de preenchimento

Tabelas retangulares

Figuras

Miniaplicativos, mapas etc.

Informações extras

Semelhante a [TD]<a>[TN]

Ponto de partida de URL

Exemplos de tags

body, head, html, title

br, code, dfn, em, *hn*, kbd, p, strong

a

dl, dt, dd, ol, ul, li

form, input, label, option, textarea

caption, table, td, th, tr

img

object, param

meta

link

base

[F]Figura 7.56

[FL] Os módulos e as tags de XHTML Basic

Apesar do acordo sobre o uso da XHTML Basic, paira uma ameaça ao WAP e ao i-mode: o padrão 802.11. A Web sem fio de segunda geração deve funcionar a 384 kbps, bem melhor que os 9600 bps da primeira geração, mas muito pior que os 11 Mbps ou 54 Mbps oferecidos pelas redes 802.11. É claro que o padrão 802.11 ainda não está difundido; porém, à medida que mais restaurantes, hotéis, lojas, empresas, aeroportos, estações rodoviárias, museus, universidades, hospitais e outras organizações decidirem instalar estações base para seus funcionários e clientes, poderá haver cobertura suficiente em áreas urbanas para atrair pessoas dispostas a caminhar alguns blocos e se sentarem em um restaurante de fast food com capacidade para reconhecer dispositivos 802.11, a fim de tomarem uma xícara de café e enviarem uma mensagem de correio eletrônico. Habitualmente, as empresas poderão mostrar logotipos do padrão 802.11 ao lado

daqueles que mostram os cartões de crédito que são aceitos, e pela mesma razão: atrair clientes. Mapas urbanos (que possam ser transferidos por download, naturalmente) poderão mostrar as áreas cobertas em verde e as áreas sem cobertura em vermelho, para que as pessoas possam passar de uma estação base para outra, como nômades vagando de um oásis para outro no deserto.
Embora os restaurantes de fast food possam ser rápidos na instalação de estações base 802.11, os fazendeiros provavelmente não terão tanta rapidez, e portanto a cobertura será pontual e limitada às áreas urbanas das cidades, devido ao alcance limitado das redes 802.11 (no máximo, algumas centenas de metros). Isso poderá levar a dispositivos sem fios de modo dual que utilizarão o padrão 802.11 se puderem captar um sinal, voltando ao WAP se não puderem captá-lo.

[T2] 7.4 Multimídia

A Web sem fio é um novo desenvolvimento interessante, mas não é o único. Para muitas pessoas, a multimídia é o Santo Graal das redes. Quando essa palavra é mencionada, visionários e executivos entram em delírio. Os primeiros vêem imensos desafios tecnológicos em fornecer vídeos (interativos) sob demanda a todos os lares. Os outros antevêem os enormes lucros. Como a multimídia exige alta largura de banda, é difícil fazê-la funcionar sobre conexões fixas. Até mesmo vídeo com qualidade VHS em redes sem fios está distante alguns anos; assim, nosso tratamento se concentrará em sistemas fisicamente conectados.
Literalmente, multimídia quer dizer apenas dois ou mais meios físicos. Se quisesse se juntar à febre da multimídia, a editora deste livro poderia anunciá-lo como uma obra que utiliza tecnologia de multimídia. Afinal, o livro é apresentado através de dois meios distintos: textos e gráficos (as figuras). Entretanto, quando as pessoas se referem à multimídia, em geral elas querem se referir à combinação de dois ou mais **meios contínuos**, ou seja, meios que têm de ser reproduzidos

durante um intervalo de tempo bem definido, e em geral com alguma interação do usuário. Na prática, os dois meios normalmente são áudio e vídeo, isto é, sons e imagens em movimento.

Entretanto, muitas pessoas se referem com freqüência a áudio puro, como telefonia da Internet ou rádio da Internet como multimídia, o que claramente não é verdade. Na realidade, um termo melhor é **mídia de fluxo**, mas seguiremos a tendência e também consideraremos o áudio em tempo real uma forma de multimídia. Nas próximas seções, examinaremos as maneiras como os computadores processam áudio e vídeo, como eles são compactados, e também algumas aplicações de rede dessas tecnologias. Para ver um tratamento completo (em três volumes) da multimídia em rede, consulte (Steinmetz e Nahrstedt, 2002; Steinmetz e Nahrstedt, 2003a; e Steinmetz e Nahrstedt 2003b).

[T3] 7.4.1 Introdução ao áudio digital

Um sinal de áudio (som) é uma onda acústica unidimensional (de pressão). Quando uma onda acústica entra no ouvido, o tímpano vibra, fazendo com que os minúsculos ossos do ouvido interno vibrem também, enviando impulsos nervosos ao cérebro. Esses pulsos são percebidos como sons pelo ouvinte. Da mesma forma, quando uma onda acústica chega a um microfone, o microfone gera um sinal elétrico, representando a amplitude do som como uma função do tempo. A representação, o processamento, o armazenamento e a transmissão desses sinais de áudio formam a área de maior interesse do estudo dos sistemas de multimídia.

A faixa de freqüências sonoras captadas pelo ouvido humano começa em 20 e chega a 20.000 Hz. Alguns animais, em especial os cães, podem ouvir freqüências mais altas. Os ouvidos percebem os sons no modo logarítmico; portanto, a razão de dois sons com potência A e B é, por convenção, expressa em

dB (decibéis) de acordo com a fórmula:

$$dB = 10 \log_{10}(A/B)$$

Se definirmos o limite mínimo de audibilidade (uma pressão de cerca de 0,0003 dina/cm²) para uma onda senoidal de 1 kHz como 0 dB, uma conversa normal terá cerca de 50 dB, e o limite máximo tolerável será de aproximadamente 120 dB, um intervalo dinâmico da ordem de um milhão.

O ouvido humano é supreendentemente sensível a variações de som que duram apenas milésimos de segundo. O olho, ao contrário, não percebe mudanças no nível da luz que durem por menos de alguns milissegundos. O resultado dessa observação é que a flutuação de apenas alguns milésimos de segundo durante uma transmissão de multimídia afeta mais a qualidade do som percebido do que a qualidade da imagem percebida.

Os sinais de áudio podem ser convertidos para a forma digital por um **ADC (Analog Digital Converter)**. Um ADC recebe um sinal elétrico como entrada e gera um número binário como saída. Na Figura 7.57(a) podemos ver um exemplo de uma onda senoidal. Para representar esse sinal na forma digital, podemos obter amostras a cada ΔT segundos, como indicam as barras da Figura 7.57(b). Se um sinal sonoro não for uma onda senoidal pura mas sim uma superposição linear de ondas senoidais, onde o componente de freqüência mais alta presente é f, o teorema de Nyquist (ver Capítulo 2) diz que é suficiente obter amostras a uma freqüência $2f$. Usar uma taxa de amostragem maior não é interessante, pois as freqüências mais altas que essa amostragem poderia detectar não estão presentes.

[arte: ver original p. 675]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 7.57

[FL] (a) Uma onda senoidal. (b) Amostragem da onda senoidal. (c) Quantização das amostras para 4 bits

As amostras digitais nunca são exatas. As amostras da Figura 7.57(c) permitem apenas nove valores, de -1,00 até + 1,00 em intervalos de 0,25. Uma amostra de 8 bits permitiria 256 valores distintos. Uma amostra de 16 bits permitiria 65.536 valores distintos. O erro introduzido pelo número finito de bits por amostra é chamado **ruído de quantização.** Se ele for muito grande, o ouvido poderá detectá-lo.

Dois exemplos bem conhecidos em que a amostragem do som é usada são os telefones e os CDs de áudio. A modulação de código de pulso, como era usada no sistema telefônico, utiliza amostras de 8 bits geradas 8000 vezes por segundo. Na América do Norte e no Japão, 7 bits correspondem a dados e ´1 é para controle. Na Europa, todos os 8 bits são usados para dados. Esse sistema proporciona uma taxa de dados de 56.000 ou 64.000 bps. Com apenas 8000 amostras por segundo, as freqüências acima de 4 kHz são perdidas.

Os CDs convencionais (de áudio) são digitais com uma taxa de amostragem de 44.100 amostras por segundo, o suficiente para capturar freqüências de até 22.050 Hz, o que é bom para os seres humanos mas ruim para os cães. As amostras têm 16 bits cada e são lineares na faixa de amplitudes. Observe que as amostras de 16 bits só permitem 65.536 valores distintos, embora a amplitude dinâmica do ouvido seja de cerca de 1 milhão, quando medida em intervalos a partir do mais baixo som audível. Portanto, o uso de apenas 16 bits por amostra introduz algum ruído de quantização (ainda que a amplitude dinâmica total não seja utilizada — os CDs não foram feitos para prejudicar a audição das pessoas). Com 44.100 amostras por segundo de 16 bits cada, um CD convencional precisa

de uma largura de banda de 705,6 kbps para som monofônico e 1,411 Mbps para som estéreo. Embora essa faixa seja menor do que o vídeo necessita (veja a seguir), ainda é necessário quase um canal T1 completo para transmitir arquivos de som não compactados com qualidade de CD estéreo em tempo real.

O som digitalizado pode ser facilmente processado por computadores através de software. Existem dezenas de programas para PCs que permitem aos usuários gravar, reproduzir, editar, mixar e armazenar sinais de áudio de diversas origens. Praticamente todas as gravações e edições de som profissional são digitais hoje em dia.

É claro que a música é apenas um caso especial de recursos de áudio, mas é muito importante. Outro caso especial importante é a voz. A fala humana tende a ter uma amplitude de 600 a 6000 Hz. A fala é composta de vogais e consoantes, que têm propriedades diferentes. As vogais são produzidas quando o trato vocal está desobstruído, emitindo ressonâncias cuja freqüência fundamental depende do tamanho e da forma do sistema vocal e da posição da língua e da mandíbula do orador. Esses sons são praticamente periódicos, com intervalos de cerca de 30 milissegundos. As consoantes são produzidas quando o trato vocal está parcialmente bloqueado. Esses sons são menos freqüentes que as vogais.

Alguns sistemas de geração e transmissão da fala utilizam modelos do sistema vocal para reduzir a fala a alguns parâmetros (por exemplo, o tamanho e a forma de diversas cavidades), em vez de fazer apenas uma amostragem de uma forma de onda da fala. Porém, o modo como esses codificadores vocais funcionam está além do escopo deste livro.

[T3] 7.4.2 Compactação de áudio

O áudio com qualidade de CD requer uma largura de banda de transmissão de 1,411 Mbps, conforme acabamos de ver. É claro que é necessária uma

compactação significativa para tornar prática a transmissão pela Internet. Por essa razão, foram desenvolvidos vários algoritmos de compactação de áudio. Talvez o mais popular seja o áudio MPEG, que tem três camadas (variantes), das quais o **MP3 (MPEG audio layer 3)** é o mais eficiente e conhecido. Grandes quantidades de música em formato MP3 estão disponíveis na Internet, nem todas legais, o que resultou em numerosos processos dos artistas e dos detentores dos direitos autorais. O MP3 pertence à porção de áudio do padrão de compactação de vídeo MPEG. Discutiremos a compactação de vídeo mais adiante neste capítulo; agora, vamos examinar a compactação de áudio.

A compactação de áudio pode ser feita de duas maneiras. Na **codificação de forma de onda**, o sinal é transformado matematicamente por uma transformação de Fourier em seus componentes de freqüência. A Figura 2.1(a) mostra um exemplo de função de tempo e suas amplitudes de Fourier. A amplitude de cada componente é então codificada de modo mínimo. O objetivo é reproduzir com precisão a forma de onda na outra extremidade, com a menor quantidade de bits possível.

O outro modo, chamado **codificação perceptiva**, explora certas falhas no sistema auditivo humano para codificar um sinal que soe da mesma forma para um ouvinte humano, ainda que pareça bem diferente em um osciloscópio. A codificação perceptiva se baseia na ciência da **psicoacústica** — a forma como as pessoas percebem o som. O MP3 é baseado na codificação perceptiva.

A principal propriedade da codificação perceptiva é que alguns sons podem **mascarar** outros sons. Imagine que você esteja transmitindo um concerto de flauta ao vivo em um dia quente de verão. De repente, um equipe de trabalhadores liga suas britadeiras e começa a esburacar a rua. Ninguém mais consegue ouvir a flauta. Seu som foi mascarado pelas britadeiras. Para fins de transmissão, agora é suficiente codificar apenas a banda de freqüência usada

pelas britadeiras, porque os ouvintes não podem mais ouvir a flauta. Isso se chama **mascaramento de freqüência** — a habilidade de um som em alto volume em uma banda de freqüência ocultar um som mais suave em outra banda de freqüência que seria audível na ausência do som alto. De fato, mesmo depois que as britadeiras pararem de funcionar, a flauta será inaudível por um curto período de tempo, porque o ouvido diminuiu seu ganho quando elas começarem a funcionar e o ouvido demora um tempo finito para aumentar novamente o ganho. Esse efeito é chamado **mascaramento temporal**.

Para tornar esses efeitos mais quantitativos, imagine a experiência 1. Uma pessoa em um quarto silencioso coloca fones de ouvido conectados à placa de som de um computador. O computador gera uma onda senoidal pura a 100 Hz em potência baixa, mas gradualmente crescente. A pessoa é instruída a pressionar uma tecla ao ouvir o tom. O computador registra o nível de potência atual e depois repete a experiência a 200 Hz, 300 Hz e em todas as outras freqüências, até o limite da audição humana. Quando é calculada a média para várias pessoas, o resultado é um grafo logarítmico mostrando a potência necessária para que um tom seja audível, semelhante ao grafo da Figura 7.58(a). Uma conseqüência direta dessa curva é que nunca se torna necessário codificar quaisquer freqüências cuja potência fique abaixo do limiar de audibilidade. Por exemplo, se a potência a 100 Hz fosse 20 dB na Figura 7.58(a), ela poderia ser omitida da saída sem qualquer perda perceptível de qualidade, porque 20 dB a 100 Hz fica abaixo do nível de audibilidade.

Agora considere a experiência 2. O computador executa a experiência 1 novamente, mas dessa vez com uma onda senoidal de amplitude constante, digamos, de 150 Hz, superposta à freqüência de teste. Descobrimos que o limiar de audibilidade para freqüências próximas a 150 Hz é elevado, como mostra a Figura 7.58(b).

Uma conseqüência dessa nova observação é que, mantendo-se o controle dos sinais que estão sendo mascarados por sinais mais potentes em bandas de freqüência próximas, podemos omitir cada vez mais freqüências no sinal codificado, economizando bits. Na Figura 7.58, o sinal de 125 Hz pode ser completamente omitido da saída e ninguém será capaz de perceber a diferença. Mesmo depois que um sinal potente for interrompido em alguma banda de freqüência, o conhecimento de suas propriedades de mascaramento temporal nos permitirá continuar a omitir as freqüências mascaradas durante algum intervalo de tempo, enquanto o ouvido se recupera. A essência do MP3 é efetuar a transformação de Fourier do som para obter a potência em cada freqüência, e depois transmitir apenas as freqüências não mascaradas, codificando-as na menor quantidade de bits possível.

[arte: ver original p. 678]

[Dísticos]

[1] Potência (dB)

[2] Limiar de audibilidade

[3] Freqüência (kHz)

(a)

[4] Potência (dB)

[5] Sinal mascarado

[6] Mascaramento de sinal a 150 Hz

[7] Limiar de audibilidade

[8] Freqüência (kHz)

(b)

[F]Figura 7.58

[FL] (a) O limiar de audibilidade como uma função de freqüência. (b) O efeito de mascaramento

Com essas informações, agora podemos ver como a codificação é feita. A compactação de áudio é realizada fazendo-se a amostragem da forma de onda a 32 kHz, a 44,1 kHz ou a 48 kHz. A amostragem pode ser feita em um ou dois canais, em qualquer uma dessas quatro configurações:

1. Monofônica (um único fluxo de entrada).
2. Monofônica dual (por exemplo, uma rilha sonora em inglês e outra em japonês).
3. Estéreo disjunto (cada canal compactado separadamente).
4. Estéreo conjunto (redundância entre canais completamente explorada).

Primeiro, é escolhida a taxa de bits de saída. O MP3 pode compactar um CD de rock 'n roll até 96 kbps com pouca perda de qualidade perceptível, mesmo para os fãs de rock 'n roll sem qualquer deficiência auditiva. Para um concerto de piano, são necessários pelo menos 128 kbps. As taxas são diferentes, porque a relação sinal/ruído do rock 'n roll é muito mais alta que a de um concerto de piano (no sentido de engenharia). Também é possível escolher taxas de saída mais baixas e aceitar uma certa perda de qualidade.

Em seguida, as amostras são processadas em grupos de 1152 (cerca de 26 ms). Primeiro, cada grupo passa por 32 filtros digitais para gerar 32 bandas de freqüência. Ao mesmo tempo, a entrada é entregue a um modelo psicoacústico para se determinar as freqüências mascaradas. Depois, cada uma das 32 bandas de freqüência é transformada ainda mais para fornecer uma resolução espectral mais precisa.

Na fase seguinte, a quantidade de bits disponível é dividida entre as bandas, com mais bits alocados às bandas com a maior potência espectral não mascarada, menos bits alocados a bandas não mascaradas com menor potência espectral e nenhum bit alocado a bandas mascaradas. Finalmente, os bits são codificados

com o emprego da codificação de Huffman, que atribui códigos curtos a números que aparecem com freqüência e códigos longos aos que ocorrem com pouca freqüência.

Na realidade, há outros detalhes. Também são usadas diversas técnicas para redução de ruído, anti-serrilhado e exploração da redundância entre canais, se possível, mas essas técnicas estão além do escopo deste livro. Uma introdução matemática mais formal ao processo é dada em (Pan, 1995).

[T3] 7.4.3 Áudio de fluxo

Agora, vamos passar da tecnologia de áudio digital para três de suas aplicações em redes. A primeira é o áudio de fluxo (streaming audio), isto é, a audição de sons pela Internet. Essa aplicação também é chamada música por demanda. Nas duas seções seguintes, estudaremos o rádio da Internet e a voz sobre IP, respectivamente.

A Internet está repleta de Web sites de música, muitos deles com listas de títulos de canções nos quais os usuários podem clicar para ouvir as canções. Alguns deles são sites gratuitos (por exemplo, novas bandas procurando publicidade); outros exigem o pagamento em troca da música, embora com freqüência também ofereçam algumas amostras gratuitas (por exemplo, os primeiros 15 segundos de uma canção). A maneira mais simples de reproduzir a música é ilustrada na Figura 7.59.

[arte: ver original p. 679]

[Dísticos]

[1] Máquina cliente

Reprodutor de mídia Navegador

6 5

Disco

[2] Máquina servidora

1, 2 Servidor da Web

4 3

Disco

[3]

1. Estabelece conexão TCP

2. Envia solicitação GET de HTTP

3. O servidor obtém o arquivo do disco

4. Arquivo devolvido

5. O navegador grava o arquivo no disco

6. O reprodutor de mídia vai buscar o arquivo bloco por bloco e o reproduz

[F]Figura 7.59

[FL] Um modo simples de implementar música reproduzida através de cliques em uma página da Web

O processo começa com um clique do usuário em uma canção. Em seguida, o navegador entra em ação. Na etapa 1 ele estabelece uma conexão TCP para o servidor da Web ao qual a canção está ligada por um hiperlink. A etapa 2 é o envio de uma solicitação *GET* em HTTP para obter a canção. Em seguida (etapas 3 e 4), o servidor vai buscar a canção (que é apenas um arquivo em MP3 ou algum outro formato) no disco e o envia de volta ao navegador. Se o arquivo for maior que a memória do servidor, ele poderá buscar e enviar a música um bloco por vez.

Usando o tipo MIME, por exemplo, *audio/mp3* (ou a extensão de arquivo), o navegador descobre como deve exibir o arquivo. Normalmente, haverá uma aplicação auxiliar como RealOne Player, Windows Media Player ou Winamp, associada a esse tipo arquivo. Tendo em vista que a maneira habitual do

navegador se comunicar com um auxiliar é gravar o conteúdo em um arquivo de rascunho, primeiro ele irá gravar todo o arquivo de música como um arquivo de rascunho no disco (etapa 5). Em seguida, ele iniciará o reprodutor de mídia e passará a ele o nome do arquivo de rascunho. Na etapa 6, o reprodutor de mídia começará a buscar e reproduzir a música, bloco por bloco.

Em princípio, essa abordagem é completamente correta e reproduzirá a música. O único problema é que a canção inteira tem de ser transmitida pela rede antes que a música possa começar. Se a canção tiver 4 MB (um tamanho típico para uma canção de MP3) e o modem for de 56 kbps, o usuário será presenteado com quase 10 minutos de silêncio enquanto a canção está sendo baixada. Nem todos os apreciadores de música gostam dessa idéia, em especial porque a próxima canção também começará com 10 minutos de tempo de download, e a seguinte também.

Para contornar esse problema sem alterar o funcionamento do navegador, os sites de música elaboraram o esquema a seguir. O arquivo vinculado ao título da canção não é o arquivo de música real. Em vez disso, ele é o que se denomina um **metarquivo**, um arquivo muito curto que simplesmente identifica a música. Um metarquivo típico poderia ter apenas uma linha de texto ASCII e seria semelhante a:

[TD]rtsp://joes-audio-server/song-0025.mp3[TN]

Quando o navegador obtém o arquivo de 1 linha, ele o grava no disco em um arquivo de rascunho, inicia o reprodutor de mídia como um auxiliar e entrega a ele o nome do arquivo de rascunho, como sempre. O reprodutor de mídia lê o arquivo e vê que ele contém um URL. Em seguida, entra em contato com *joes-audio-server* e pede a canção. Observe que o navegador não está mais no loop.

Na maioria dos casos, o servidor identificado no metarquivo não é o mesmo servidor da Web. De fato, em geral ele nem sequer é um servidor HTTP, mas sim

um servidor de mídia especializado. Nesse exemplo, o servidor de mídia utiliza o **RTSP (Real Time Streaming Protocol)**, como indica o nome do esquema *rtsp*. Ele é descrito na RFC 2326.

O reprodutor de mídia tem quatro tarefas importantes a cumprir:

1. Administrar a interface do usuário.

2. Lidar com erros de transmissão.

3. Descompactar a música.

4. Eliminar a flutuação.

A maioria dos reprodutores de mídia atuais tem uma interface do usuário vistosa, às vezes simulando uma unidade estéreo, com botões, controles deslizantes e visores. Com freqüência, existem painéis frontais intercambiáveis, chamados **skins**, que o usuário pode colocar no reprodutor. O reprodutor de mídia tem de administrar tudo isso e interagir com o usuário.

Seu segundo trabalho é lidar com erros. A transmissão de música em tempo real raras vezes utiliza o TCP, porque um erro e um retransmissão poderiam introduzir um intervalo inaceitável na música. Em vez disso, a transmissão real costuma ser feita com um protocolo como o RTP, que estudamos no Capítulo 6. Como a maioria dos protocolos de tempo real, o RTP é disposto em camadas sobre o UDP, e portanto pode ocorrer perda de pacotes. Cabe ao reprodutor lidar com isso.

Em alguns casos, a música é intercalada para facilitar o tratamento de erros. Por exemplo, um pacote poderia conter 220 amostras estereofônicas, cada uma contendo um par de números de 16 bits, normalmente boa para 5 ms de música. No entanto, o protocolo poderia enviar todas as amostras ímpares correspondentes a um intervalo de 10 ms em um pacote e todas as amostras pares no pacote seguinte. Então, um pacote perdido não representaria um intervalo de 5 ms na música, mas sim a perda de amostras alternadas durante 10

ms. Essa perda pode ser manipulada com facilidade fazendo-se o reprodutor de mídia realizar a interpolação, usando a amostra anterior e a seguinte, a fim de estimar o valor perdido.

O uso da intercalação para alcançar a recuperação de erros é ilustrado na Figura 7.60. Aqui cada pacote contém amostras de tempos alternados para um intervalo de 10 ms. Conseqüentemente, a perda do pacote 3, como mostra a figura, não cria um intervalo na música, mas apenas diminui a resolução temporal para algum intervalo. Os valores perdidos podem ser interpolados para fornecer música contínua. Esse esquema específico só funciona com amostragem não compactada, mas demonstra como a codificação inteligente pode converter um pacote perdido em qualidade de áudio mais baixa, em vez de transformá-lo em um intervalo de tempo. No entanto, a RFC 3119 fornece um esquema que funciona com áudio compactado.

[arte: ver original p. 681]

[Dísticos]

[1] Este pacote contém 220 amostras de tempos pares

(a)

Pacote 0 1 2

[2] Este pacote contém 220 amostras de tempos ímpares

Perdido

4 5

[3] Legenda

Amostra de tempo par

Amostra de tempo ímpar

[4] (b)

0 5 10 15 20 25 30

Tempo (ms)

[F]Figura 7.60

[FL] Quando os pacotes transportam amostras alternadas, a perda de um pacote reduz a resolução temporal, em vez de criar um intervalo no tempo

O terceiro trabalho do reprodutor de mídia é descompactar a música. Embora essa tarefa seja intensa do ponto de vista da computação, ela é bastante simples. O quarto trabalho é eliminar a flutuação, a ruína de todos os sistemas de tempo real. Todos os sistemas de áudio de fluxo começam armazenando no buffer cerca de 10 a 15 s de música antes de iniciarem a reprodução, como mostra a Figura 7.61. No caso ideal, o servidor continuará a preencher o buffer na taxa exata em que ele está sendo drenado pelo reprodutor de mídia mas, na realidade, isso pode não acontecer, e assim talvez seja útil um feedback no loop.

Podem ser usadas duas abordagens para manter o buffer abastecido. Com um **servidor pull**, desde que exista espaço no buffer para outro bloco, o reprodutor de mídia continuará a transmitir solicitações ao servidor, em busca de um bloco adicional. Seu objetivo é manter o buffer o mais cheio possível.

[arte: ver original p. 682]

[Dísticos]

[1] Máquina cliente

Buffer

Reprodutor de mídia

Marca de nível baixo Marca de nível alto

[2] Máquina servidora

Servidor de mídia

[F]Figura 7.61

[FL] O reprodutor de mídia armazena em buffers a entrada do servidor de mídia e reproduz a música a partir do buffer, em vez de reproduzi-la diretamente da rede

A desvantagem de um servidor pull é a quantidade de solicitações de dados desnecessárias. O servidor sabe que enviou o arquivo inteiro; então, por que o reprodutor continua a pedir partes do arquivo? Por essa razão, raramente ele é usado.

Com um **servidor push**, o reprodutor de mídia envia uma solicitação *PLAY* e o servidor simplesmente continua a empurrar dados. Nesse caso, há duas possibilidades: o servidor de mídia funciona em velocidade normal de reprodução ou mais rápido. Em ambos os casos, alguns dados são armazenados no buffer antes do início da reprodução. Se o servidor funcionar em velocidade de reprodução normal, os dados que chegarem dele serão anexados ao final do buffer, e o reprodutor removerá dados do início do buffer para reprodução. Desde que tudo funcione perfeitamente, o volume de dados no buffer permanecerá constante no tempo. Esse esquema é simples, porque nenhuma mensagem de controle é exigida em um ou outro sentido.

O outro esquema de push é fazer o servidor bombear dados com maior rapidez que. Nesse caso, se o servidor não puder ter a garantia de funcionar a uma velocidade regular, ele terá a oportunidade de aumentar sua velocidade se ficar para trás. Entretanto, aqui existe um grande problema: podem ocorrer estouros de buffer se o servidor entregar dados com maior rapidez do que eles são consumidos (e ele terá de ser capaz de fazer isso para evitar intervalos).

A solução é o reprodutor de mídia definir uma **marca de nível baixo** e uma **marca de nível alto** no buffer. Basicamente, o servidor bombeia os dados para que o buffer seja preenchido até a marca de nível alto. Em seguida, o reprodutor de mídia solicita que ele faça uma pausa. Tendo em vista que os dados continuarão a ser despejados até o servidor receber a solicitação de pausa, a distância entre a marca de nível alto e o fim do buffer terá de ser maior que o produto da largura

de banda pelo retardo da rede. Depois que o servidor tiver parado, o buffer começará a se esvaziar. Quando ele chegar à marca de nível baixo, o reprodutor de mídia pedirá ao servidor de mídia que comece outra vez. A marca de nível baixo tem de estar posicionada de tal forma que não ocorra o esgotamento (underrun) do buffer.

Para operar um servidor push, um reprodutor de mídia precisa de um controle remoto para ele. É isso que o RTSP oferece. Ele é definido na RFC 2326 e fornece o mecanismo para o reprodutor controlar o servidor. Ele não estabelece o fluxo de dados, que em geral é o RTP. Os principais comandos fornecidos pelo RTSP estão listados na Figura 7.62.

[arte: ver original p. 683]

[T]Tabela

Comando	**Ação do servidor**
DESCRIBE	Lista de parâmetros de mídia
SETUP	Estabelece um canal lógico entre o reprodutor e o servidor
PLAY	Começa a enviar dados para o cliente
RECORD	Começa a aceitar dados do cliente
PAUSE	Interrompe temporariamente o envio de dados
TEARDOWN	Libera o canal lógico

[F]Figura 7.62

[FL] Comandos RTSP do reprodutor para o servidor

[T3] 7.4.4 Rádio da Internet

Depois que se tornou possível transmitir áudio de fluxo pela Internet, as estações de rádio comerciais captaram a idéia de transmitir por difusão seu conteúdo pela Internet, bem como pelo ar. Não faz muito tempo que as estações de rádio dos estudantes começaram a emitir seus sinais pela Internet. Em seguida, os

estudantes de faculdades começaram a criar suas próprias estações de rádio. Com a tecnologia atual, praticamente qualquer pessoa pode dar início a uma estação de rádio. todo o tema de rádio pela Internet é muito novo e se encontra em estado de constante mudança, mas vale a pena examinarmos alguns detalhes sobre ele.

Existem duas abordagens gerais para rádio da Internet. Na primeira, os programas são gravados previamente e armazenados em disco. Os ouvintes podem se conectar aos arquivos de armazenamento da estação de rádio, acessar qualquer programa, baixá-lo e ouvi-lo. De fato, isso é exatamente igual ao áudio de fluxo que acabamos de estudar. Também é possível armazenar cada programa logo após sua transmissão ao vivo, de forma que o arquivo de armazenamento só esteja atualizado, digamos, durante meia hora ou menos depois da transmissão ao vivo. As vantagens dessa estratégia são a facilidade, o fato de todas as técnicas que estudamos também funcionarem aqui e a possibilidade dos ouvintes escolherem entre todos os programas que formam o arquivo.

A outra estratégia é a transmissão ao vivo pela Internet. Algumas estações transmitem pelo ar e pela Internet ao mesmo tempo, mas há um número cada vez maior de estações de rádio que só funcionam pela Internet. Algumas técnicas aplicáveis ao áudio de fluxo se aplicam igualmente à transmissão de rádio ao vivo pela Internet, embora também existam algumas diferenças importantes.

Um ponto comum é a necessidade de armazenamento em buffer no lado do usuário para suavizar a flutuação. Reunindo-se 10 ou 15 segundos de transmissão de rádio antes de iniciar a reprodução, o áudio pode continuar suavemente, mesmo diante de uma flutuação significativa na rede. Desde que todos os pacotes cheguem antes de serem necessários, não importa o momento em que cheguem.

Uma diferença fundamental é que o áudio de fluxo pode ser transmitido a uma

taxa maior que a taxa de reprodução, pois o receptor pode interrompê-lo quando a marca de nível alto for alcançada. Potencialmente, isso proporciona o tempo necessário para retransmitir pacotes perdidos, embora essa estratégia não seja muito usada. Em contraste, o rádio ao vivo está sempre sendo transmitido na velocidade exata em que é gerado e reproduzido.

Outra diferença é que uma estação de rádio ao vivo normalmente tem centenas ou milhares de ouvintes simultâneos, enquanto o áudio de fluxo é ponto a ponto. Sob essas circunstâncias, o rádio da Internet deve usar a multidifusão com os protocolos RTP/RTSP. Sem dúvida, essa é a forma mais eficiente de operar.

Na prática, o rádio da Internet não funciona assim hoje em dia. Na realidade, o usuário estabelece uma conexão TCP para a estação e o conteúdo é enviado pela conexão TCP. É claro que isso cria diversos problemas, como a interrupção de fluxo quando a janela está cheia, pacotes perdidos que chegam ao timeout e são retransmitidos e assim por diante.

Há três razões para o uso da unidifusão por TCP em vez da multidifusão por RTP. Primeiro, poucos ISPs admitem multidifusão, e portanto essa não é uma opção prática. Em segundo lugar, o RTP é menos bem conhecido que o TCP, e freqüentemente as estações de rádio além de pequenas têm pouca experiência em informática, e assim é bem mais fácil usar um protocolo bastante reconhecido e admitido por todos os pacotes de software. Em terceiro lugar, muitas pessoas escutam rádio da Internet no trabalho; na prática, isso significa que a transmissão tem de atravessar um firewall. A maioria dos administradores de sistemas configuram seu firewall para proteger sua LAN de visitantes indesejáveis. Em geral, eles permitem conexões TCP pela porta remota 25 (SMTP para correio eletrônico), pacotes UDP da porta remota 53 (DNS) e conexões TCP da porta remota 80 (HTTP para a Web). Quase todas as outras conexões são bloqueadas, inclusive o RTP. Desse modo, o único meio de fazer o sinal de rádio passar pelo

firewall é o Web site fingir ser um servidor HTTP, pelo menos para o firewall, e usar servidores HTTP que se comuniquem usando o TCP. Essas medidas severas, embora forneçam apenas um nível mínimo de segurança, freqüentemente forçam as aplicações de multimídia a usarem modos de operação bem menos eficiente.

Tendo em vista que o rádio da Internet é um novo meio de comunicação, as guerras de formatos estão acirradas. RealAudio, Windows Media Audio e MP3 estão competindo agressivamente nesse mercado e um deles deverá se tornar o formato dominante para rádio da Internet. Um novo concorrente é o Vorbis, tecnicamente semelhante ao MP3, mas que tem fonte aberta e é diferente o bastante para não usar as patentes em que o MP3 se baseia.

Uma estação típica de rádio da Internet tem uma página da Web listando sua programação, informações sobre seu DJs e anunciadores e muitos anúncios. Também existe um ou mais ícones listando os formatos de áudio que a estação admite (ou simplesmente ESCUTE AGORA, se houver apenas um formato possível). Esses ícones ou ESCUTE AGORA são metarquivos vinculados do tipo que descrevemos antes.

Quando um usuário clica em um dos ícones, o pequeno metarquivo é enviado. O navegador utiliza seu tipo MIME ou sua extensão de arquivo para determinar o auxiliar (isto é, o reprodutor de mídia) apropriado para o metarquivo. Em seguida, ele grava o metarquivo em um arquivo de rascunho no disco, inicia o reprodutor de mídia e entrega a ele o nome do arquivo de rascunho. O reprodutor de mídia lê o arquivo de rascunho, vê o URL que ele contém (normalmente com um esquema *http* em vez de *rtsp* para contornar o problema do firewall e porque algumas aplicações de multimídia populares funcionam desse modo), entra em contato com o servidor e começa a agir como uma estação de rádio. A propósito, o áudio só tem um fluxo, e assim o esquema *http* funciona; no caso do vídeo, que tem pelo menos dois fluxos, *http* falha e se torna realmente necessário algo como

rtsp.

Outro desenvolvimento interessante na área de rádio da Internet é uma organização em que qualquer pessoa, até mesmo um estudante, pode configurar e operar uma estação de rádio. Os principais componentes estão ilustrados na Figura 7.63. A base da estação é um PC comum com uma placa de som e um microfone. O software consiste em um reprodutor de mídia, como Winamp ou Freeamp, com um plug-in para captura de áudio e um codec correspondente ao formato de saída selecionado como, por exemplo, MP3 ou Vorbis.

[arte: ver original p. 685]

[Dísticos]

[1] Microfone

[2] Plug-in de captura de áudio

[3] Reprodutor de mídia

[4] Plug-in de codec

[5] Internet

[6] Servidor de mídia

[7] Conexões TCP para ouvintes

[8] PC do estudante

[F]Figura 7.63

[FL] Uma estação de rádio estudantil

O fluxo de áudio gerado pela estação é então transmitido pela Internet para um grande servidor, que lida com a distribuição desse fluxo para um grande número de conexões TCP. Em geral, o servidor admite muitas estações pequenas, e também mantém um diretório das estações que tem e do programa que está no ar em cada uma. Os ouvintes potenciais vão ao servidor, selecionam uma estação e recebem um fluxo TCP. Existem pacotes de software comerciais para

administrar todos os itens, além de pacotes de fonte aberta como o icecast. Também existem servidores dispostos a lidar com a distribuição mediante o pagamento de uma taxa.

[T3] 7.4.5 Voz sobre IP

Era uma vez um sistema público de telefonia comutada usado principalmente para tráfego de voz com um pouco de tráfego de dados aqui e ali. Porém, o tráfego de dados cresceu e cresceu e, por volta de 1999, o número de bits de dados transferidos igualou o número de bits de voz (pois a voz está codificada em PCM nos troncos, e assim pode ser medida em bits/s). Em 2002, o volume do tráfego de dados era dez vezes maior que o volume do tráfego de voz, e ainda continua a crescer exponencialmente, enquanto o tráfego de voz permanece quase no mesmo nível (crescendo 5% ao ano).

Como conseqüência desses números, muitas operadoras de redes de comutação de pacotes de repente ficaram interessadas em transportar voz sobre suas redes de dados. O volume de largura de banda adicional exigida para voz é minúsculo, pois as redes de pacotes são dimensionadas para o tráfego de dados. No entanto, a conta telefônica de um consumidor médio provavelmente é maior que sua conta da Internet, e assim as operadoras de redes de dados viram na telefonia da Internet um modo de ganhar um bom dinheiro extra sem terem de instalar sequer um novo cabo de fibra. Desse modo, nasceu a **telefonia da Internet** (também conhecida como **voz sobre IP**).

[T4] H.323

Ficou claro para todos desde o início que, se cada fornecedor projetasse sua própria pilha de protocolos, o sistema nunca funcionaria. Para evitar esse problema, várias partes interessadas se reuniram sob o patrocínio da ITU para

desenvolver padrões. Em 1996, a ITU emitiu a recomendação **H.323**, intitulada "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service" (ou seja, sistemas e equipamentos de telefonia visual para redes locais que oferecem uma qualidade de serviço não garantida). Só mesmo a indústria de telefonia pensaria em tal título. A recomendação foi revisada em 1998, e essa recomendação H.323 revisada foi a base para os primeiros sistemas amplamente difundidos de telefonia da Internet. A recomendação H.323 é mais uma avaliação da arquitetura de telefonia da Internet que um protocolo específico. Ela faz referência a um grande número de protocolos específicos para codificação de voz, configuração de chamadas, sinalização, transporte de dados e outras áreas, em vez de especificar propriamente cada um desses elementos. O modelo geral é representado na Figura 7.64. No centro há um **gateway** que conecta a Internet à rede de telefonia. Ele se comunica por meio dos protocolos H.323 no lado da Internet e dos protocolos PSTN no lado da rede telefônica. Os dispositivos de comunicação são chamados **terminais.** Uma LAN pode ter um @@@**gatekeeper** (guardião) que controla os pontos extremos sob sua jurisdição, denominada **zona.**

[arte: ver original p. 686]

[Dísticos]

[1] Zona

Gatekeeper

[2] Terminal

[3] Internet

[4] Gateway

[5] Rede de telefonia

[F]Figura 7.64

[FL] O modelo arquitetônico do H.323 para telefonia da Internet

Uma rede de telefonia necessita de vários protocolos. Para começar, existe um protocolo para codificação e decodificação de voz. O sistema PCM que estudamos no Capítulo 2 é definido na recomendação da ITU **G.711**. Ele codifica um único canal de voz realizando a amostragem 8000 vezes por segundo com amostras de 8 bits, a fim de fornecer voz descompactada a 64 kbps. Todos os sistemas H.323 devem ter suporte para a G.711. No entanto, também são permitidos (mas não exigidos) outros protocolos de compactação de voz. Eles empregam diversos algoritmos de compactação e admitem diferentes compromissos entre qualidade e largura de banda. Por exemplo, a **G.723.1** aceita um bloco de 240 amostras (30 ms de voz) e utiliza a codificação profética com o objetivo de reduzi-las a 24 bytes ou a 20 bytes. Esse algoritmo oferece uma taxa de saída de 6,4 kbps ou 5,3 kbps (fatores de compactação de 10 e 12), respectivamente, com pequena perda na qualidade percebida. Outro codecs também são permitidos.

Tendo em vista que são possíveis diversos algoritmos de compactação, é necessário um protocolo para permitir que os terminais negociem o algoritmo que vão usar. Esse protocolo é chamado **H.245**. Ele também negocia outros aspectos da conexão, como a taxa de bits. O RTCP é necessário para controlar os canais do RTP. Também é preciso um protocolo para estabelecer e encerrar conexões, fornecer tons de discagem, gerar sons de chamada e o restante da telefonia padrão. A ITU **Q.931** é usada aqui. Os terminais precisam de um protocolo para se comunicar com o gatekeeper (se presente). Para esse propósito, é usado o **H.225**. O canal do PC para o gatekeeper que ele gerencia é chamado canal **RAS (Registration/Admission/Status)**. Esse canal permite que os terminais entrem e saiam da zona, solicitem e retornem largura de banda, e forneçam atualizações de status, entre outras coisas. Por fim, é necessário um protocolo para a transmissão de dados reais. O RTP é usado com esse propósito. Ele é

gerenciado pelo RTCP, como sempre. O posicionamento de todos esses protocolos é mostrado na Figura 7.65.

[arte: ver original p. 687]

[Dísticos]

[1] Voz Controle

[2] G.7xx RTCP H.225 (RAS) Q.931 (Sinalização de chamadas) H.245 (Controle de chamadas)

RTP

[3] UDP TCP

[4] IP

[5] Protocolo de enlace de dados

[6] Protocolo da camada física

[F]Figura 7.65

[FL] A pilha de protocolos H.323

Para ver como esses protocolos funcionam juntos, considere o caso de um terminal de PC em uma LAN (com um gatekeeper) que chama um telefone remoto. Primeiro, o PC tem de descobrir o gatekeeper e, para isso, transmite por difusão um pacote UDP de descoberta de gatekeeper para a porta 1718. Quando o gatekeeper responde, o PC aprende o endereço IP do gatekeeper. Agora, o PC se registra com o gatekeeper, enviando a ele uma mensagem RAS em um pacote UDP. Depois que a mensagem é aceita, o PC envia ao gatekeeper uma mensagem RAS de admissão solicitando largura de banda. Só depois que a largura de banda é concedida, é possível iniciar a configuração de chamada. A idéia de solicitar largura de banda com antecedência tem a finalidade de permitir ao gatekeeper limitar o número de chamadas, a fim de evitar saturar a linha de saída, e desse modo oferecer a qualidade de serviço necessária.

Agora, o PC estabelece uma conexão TCP para o gatekeeper, a fim de iniciar a configuração de chamada. A configuração de chamada utiliza os protocolos existentes da rede telefônica, que são orientados a conexões e, portanto, o TCP é necessário. Em contraste, o sistema telefônico não tem nada semelhante ao RAS que permita aos telefones anunciarem sua presença, e assim os projetistas do H.323 ficaram livres para usar o UDP ou o TCP para RAS. Eles optaram pelo UDP por ter um overhead mais baixo.

Depois que a largura de banda é alocada, o PC pode enviar uma mensagem *SETUP* de Q.931 pela conexão TCP. Essa mensagem especifica o número do telefone que está sendo chamado (ou o endereço IP e a porta, se for uma chamada para um computador). O gatekeeper responde com uma mensagem Q.931 *CALL PROCEEDING* para confirmar o recebimento correto da solicitação. Então, o gatekeeper encaminha a mensagem *SETUP* para o gateway.

O gateway, que é metade computador e metade switch de telefonia, faz uma chamada telefônica comum para o telefone desejado (comum). A estação final ao qual o telefone está conectado faz soar o sinal do telefone chamado e também envia de volta uma mensagem *ALERT* de Q.931 para informar ao PC chamador que a chamada teve início. Quando a pessoa na outra extremidade da linha atende o telefone, a estação final envia de volta uma mensagem *CONNECT* de Q.931 para indicar ao PC que ele tem uma conexão.

Após o estabelecimento da conexão, o gatekeeper não está mais no loop, mas é claro que o gateway está. Os pacotes subseqüentes ignoram o gatekeeper e vão diretamente para o endereço IP do gateway. Nesse momento, só temos um canal livre entre as duas partes. Trata-se apenas de uma conexão da camada física para movimentação de bits, e nada mais. Nenhum dos lados conhece qualquer detalhe sobre o outro.

O protocolo H.245 é usado agora para negociar os parâmetros da chamada. Ele

usa o canal de controle H.245, que está sempre aberto. Cada lado começa anunciando seus recursos; por exemplo, se pode lidar com vídeo (o H.323 pode manipular vídeo) ou chamadas de conferência, quais codecs são aceitos etc. Depois que cada lado sabe o que o outro pode manipular, são configurados dois canais de dados unidirecionais, e também são atribuídos a cada parte um codec e outros parâmetros. Tendo em vista que cada lado pode ter um equipamento diferente, é perfeitamente possível que os codecs dos canais direto e inverso sejam diferentes. Depois de concluídas todas as negociações, o fluxo de dados pode começar a usar o RTP. Ele é gerenciado com o RTCP, que desempenha uma função importante no controle de congestionamento. Se houver vídeo presente, o RTCP irá lidar com a sincronização de áudio/vídeo. Os diversos canais são mostrados na Figura 7.66. Quando uma das partes desliga o telefone, o canal de sinalização de chamadas do Q.931 é usado para desfazer a conexão.

Quando a chamada é encerrada, o PC chamador entra mais uma vez em contato com o gatekeeper com uma mensagem RAS para liberar a largura de banda que recebeu. Ele também pode efetuar outra chamada.

Ainda não dissemos nada sobre a qualidade de serviço, embora ela seja essencial para tornar a voz sobre IP um sucesso. A razão para isso é que a QoS não está no âmbito do H.323. Se a rede subjacente for capaz de produzir uma conexão estável e livre de flutuação entre o PC chamador (por exemplo, usando as técnicas que descrevemos no Capítulo 5) e o gateway, então a QoS da chamada será boa; caso contrário, ela não será boa. A parte do telefone emprega a PCM e está sempre isenta de flutuação.

[arte: ver original p. 689]

[Dísticos]

[1] Chamador

[2] Canal de sinalização de chamadas (Q.931)

[3] Canal de controle de chamadas (H.245)

[4] Canal de encaminhamento de dados (RTP)

[5] Canal inverso de dados (RTP)

[6] Canal de controle de dados (RTCP)

[7] Chamado

[F]Figura 7.66

[FL] Canais lógicos entre o chamador e o chamado durante a realização de uma chamada

[T4] SIP — Session Initiation Protocol

O H.323 foi projetado pela ITU. Muitas pessoas na comunidade da Internet viam esse protocolo como um produto típico das empresas de telecomunicações: grande, complexo e inflexível. Conseqüentemente, a IETF estabeleceu um comitê para projetar uma forma mais simples e mais modular de utilizar voz sobre IP.

O resultado mais importante até hoje é o **SIP (Session Initiation Protocol)**, descrito na RFC 3261. Esse protocolo descreve como instalar chamadas telefônicas da Internet, videoconferências e outras conexões de multimídia. Diferente do H.323, um conjunto de protocolos completo, o SIP é um único módulo, mas foi projetado para interoperar bem com aplicações da Internet existentes. Por exemplo, ele define números de telefones como URLs, de forma que as páginas da Web possam conter esses números, permitindo que um clique em um link inicie uma ligação telefônica (da mesma forma que o esquema *mailto* permite que um clique sobre um link abra um programa para enviar uma mensagem de correio eletrônico).

O SIP pode estabelecer sessões de duas partes (ligações telefônicas comuns), sessões de várias partes (onde todos podem ouvir e falar) e sessões de multidifusão (com um transmissor e muitos receptores). As sessões podem conter

áudio, vídeo ou dados, sendo que os dados são úteis para, por exemplo, a realização de jogos em tempo real com vários participantes. O SIP cuida apenas da configuração, do gerenciamento e do encerramento de sessões. Outros protocolos, como RTP/RTCP, são usados para transporte de dados. O SIP é um protocolo da camada de aplicação e pode funcionar sobre o UDP ou o TCP.

O SIP admite uma grande variedade de serviços, inclusive localização do chamado (que pode não estar em sua máquina local) e determinação dos recursos do chamado, bem como tratamento do mecanismo de configuração e encerramento de chamadas. No caso mais simples, o SIP configura uma sessão entre o computador do chamador e o computador do chamado; portanto, vamos examinar primeiro esse caso.

Os números de telefones no SIP são representados como URLs que utilizam o esquema *sip*, por exemplo, *sip:ilse@cs.universiry.edu*, para uma usuária chamado Ilse no host especificado pelo nome DNS *cs.university.edu*. Os URLs do SIP também podem conter endereços IPv4, endereços IPv6 ou números de telefone reais.

O protocolo SIP é um protocolo de texto modelado sobre o HTTP. Uma parte envia uma mensagem em texto ASCII que consiste em um nome de método na primeira linha, seguido por linhas adicionais contendo cabeçalhos para passagem de parâmetros. Muitos cabeçalhos são tirados do MIME para permitir ao SIP interoperar com aplicações da Internet existentes. Os seis métodos definidos pela especificação do núcleo estão listados na Figura 7.67.

[arte: ver original p. 690]

[T]Tabela

Método	**Descrição**
INVITE	Solicita a inicialização de uma sessão
ACK	Confirma que uma sessão foi inicializada

BYE Solicita o término de uma sessão

OPTIONS Consulta um host sobre seus recursos

CANCEL Cancela uma solicitação pendente

REGISTER Informa um servidor de redirecionamento sobre a localização atual do usuário

[F]Figura 7.67

[FL] Os métodos do SIP definidos na especificação do núcleo

Para estabelecer uma sessão, o chamador cria uma conexão TCP com o chamado e envia uma mensagem *INVITE* sobre ela, ou então envia a mensagem *INVITE* em um pacote UDP. Em ambos os casos, os cabeçalhos da segunda linha e das linhas subseqüentes descrevem a estrutura do corpo da mensagem, que contém os recursos do chamador, tipos de mídia e formatos. Se o chamado aceitar a ligação, ele responderá com um código de resposta do tipo HTTP (um número de três dígitos usando os grupos da Figura 7.42; 200 significa aceitação). Seguindo a linha do código de resposta, o chamado também pode fornecer informações sobre seus recursos, tipos de mídia e formatos.

A conexão é feita com o uso de um handshake de três vias, de forma que o chamador responde com uma mensagem *ACK* para finalizar o protocolo e confirmar o recibimento da mensagem 200.

Qualquer das partes pode solicitar o término de uma sessão enviando uma mensagem que contém o método *BYE*. Quando o outro lado a confirmar, a sessão será encerrada.

O método *OPTIONS* é usado para consultar uma máquina sobre seus próprios recursos. Em geral, ele é usado antes de uma sessão ser iniciada, a fim de descobrir se essa máquina é capaz de se comunicar usando voz sobre IP ou se está sendo utilizado qualquer outro tipo de sessão.

O método *REGISTER* se relaciona com a habilidade do SIP para localizar e se conectar a um usuário que está longe de casa. Essa mensagem é enviada a um servidor de localização do SIP que controlar a localização de cada usuário. Esse servidor pode ser consultado mais tarde para descobrir a localização atual do usuário. A operação de redirecionamento é ilustrada na Figura 7.68. Aqui, o chamador envia a mensagem *INVITE* a um servidor proxy para ocultar o possível redirecionamento. Então, o proxy procura o usuário e envia a mensagem *INVITE* a ele. Em seguida ele atua como um relé para as mensagens subseqüentes no handshake de três vias. As mensagens *LOOKOP* e *REPLY* não fazem parte do SIP; qualquer protocolo conveniente pode ser usado, dependendo do tipo de servidor de localização usado.

[arte: ver original p. 691]

[Dísticos]

[1] Chamador

1 INVITE

6 OK

7 ACK

[2] Servidor de localização

2 LOOKUP

3 REPLY

[3] Proxy Chamado

4 INVITE

5 OK

8 ACK

[4] 9 Dados

[F]Figura 7.68

[FL] O uso de um proxy e de servidores de redirecionamento com o SIP

O SIP tem uma grande variedade de outros recursos que não descreveremos aqui, inclusive a espera de chamadas, triagem de chamadas, criptografia e autenticação. Ele também tem a habilidade de efetuar chamadas de um computador para um telefone comum, se houver um gateway apropriado disponível entre a Internet e o sistema de telefonia.

[T4] Comparação entre H.323 e SIP

O H.323 e o SIP têm muitas semelhanças, mas também apresentam algumas diferenças. Ambos permitem chamadas com dois ou com vários participantes, usando computadores e telefones como pontos extremos. Ambos admitem a negociação de parâmetros, a criptografia e os protocolos RTP/RTCP. Um resumo das semelhanças e diferenças entre eles é apresentado na Figura 7.69.

Embora os conjuntos de características sejam semelhantes, os dois protocolos diferem extensamente na filosofia. O H.323 é um padrão pesado, típico da indústria de telefonia, especificando a pilha de protocolos completa e definindo com precisão o que é permitido e o que é proibido. Essa abordagem leva a protocolos muito bem definidos em cada camada, facilitando a tarefa de interoperabilidade. O preço pago é um padrão grande, complexo e rígido, difícil de adaptar a aplicações futuras.

Ao contrário, o SIP é um protocolo típico da Internet e funciona permutando pequenas linhas de texto ASCII. É um módulo leve que interopera bem com outros protocolos da Internet, mas não muito bem com os protocolos de sinalização do sistema telefônico existente. Pelo fato do modelo de voz sobre IP da IETF ser altamente modular, ele é flexível e pode ser adaptado com facilidade a novas aplicações. A desvantagem reside nos problemas potenciais de interoperabilidade, embora estes sejam tratados em encontros freqüentes, nos

quais diferentes implementadores se reúnem para testar seus sistemas.

O protocolo de voz sobre IP é um tópico bastante atual, com boas perspectivas de êxito. Conseqüentemente, já existem vários livros sobre o assunto. Alguns exemplos são (Collins, 2001; Davidson e Peters, 2000; Kumar *et al.*, 2001; e Wright, 2001). A edição de maio/junho de 2002 da revista *Internet Computing* tem vários artigos sobre esse tema.

[arte: ver original p. 692]

[T]Tabela

Item	**H.323**	**SIP**
Projetado por	ITU	IETF
Compatibilidade com PSTN	Sim	Ampla
Compatibilidade com a Internet	Não	Sim
Arquitetura	Monolítica	Modular
Completeza	Pilha de protocolos completa	O SIP lida apenas com a configuração
Negociação de parâmetros	Sim	Sim
Sinalização de chamadas	Q.931 sobre TCP	SIP sobre TCP ou UDP
Formato de mensagens	Binário	ASCII
Transporte de mídia	RTP/RTCP	RTP/RTCP
Chamadas de vários participantes	Sim	Sim
Conferências de multimídia	Sim	Não
Endereçamento	Número de host ou telefone	URL
Término de chamadas	Explícito ou encerramento por TCP	Explícito ou por timeout
Transmissão de mensagens instantâneas	Não	Sim
Criptografia	Sim	Sim
Tamanho do documento de padrões	1400 páginas	250 páginas
Implementação	Grande e complexa	Moderada

Status Extensamente distribuído Boas perspectivas de êxito

[F]Figura 7.69

[FL] Comparação entre o H.323 e o SIP

[T3] 7.4.6 Introdução ao vídeo

Já discutimos o ouvido humano com bastante detalhes; agora, vamos estudar as características do olho humano. O olho humano tem a seguinte propriedade: quando uma imagem é projetada na retina, ela é retida por um determinado número de milissegundos antes de se apagar. Se uma seqüência de imagens for projetada a uma velocidade de 50 ou mais quadros por segundo, o olho não perceberá que está vendo imagens discretas. Todos os sistemas de vídeo (ou seja, de televisão) exploram esse princípio para produzir filmes.

[T4] Sistemas analógicos

Para entender os sistemas de vídeo, é melhor começar pela simples e antiquada televisão em preto e branco. Para representar a imagem bidimensional como uma voltagem unidimensional de uma função do tempo, a câmera varre um feixe de elétrons rapidamente no sentido horizontal da imagem e lentamente no sentido vertical, registrando a intensidade da luz durante o percurso. No final da varredura, chamada **quadro**, o feixe de elétrons sofre um retraço. Essa intensidade transformada em função do tempo é transmitida, e os receptores repetem o processo de varredura para reconstituir a imagem. O padrão de varredura usado tanto pela câmera quanto pelo receptor é ilustrado na Figura 7.70. (Cabe observar que as câmeras CCD integram a imagem em vez de efetuarem a varredura, mas algumas câmeras e todos os monitores realizam a varredura.)

Os parâmetros exatos da varredura variam de país para país. O sistema utilizado

na América do Norte, na América do Sul e no Japão tem 525 linhas de varredura, uma relação entre os eixos vertical e horizontal de 4:3 e 30 quadros por segundo. O sistema europeu tem 625 linhas de varredura, a mesma relação entre eixos de 4:3 e 25 quadros por segundo. Nos dois sistemas, algumas linhas superiores e algumas linhas inferiores não são exibidas (para obter uma imagem aproximadamente retangular nos CRTs redondos originais). Apenas 483 das 525 linhas de varredura NTSC (e 576 das 625 linhas de varredura PAL/SECAM) são exibidas. O feixe é desligado durante o retraço vertical e, portanto, muitas estações (especialmente na Europa) utilizam esse intervalo para transmitir teletexto (páginas de texto contendo notícias, previsão do tempo, esportes, cotações da bolsa etc.).

[arte: ver original p. 693]

[Dísticos]

[1] Linha de varredura

[2] O próximo campo começa aqui

[3] Linha de varredura desenhada na tela

[4] Tempo

[5] Retraço vertical

[6] Retraço horizontal

[F]Figura 7.70

[FL] O padrão de varredura usado para vídeo e televisão NTSC

Embora a velocidade de 25 quadros por segundo seja suficiente para captar movimentos suaves, a essa taxa de quadros, muitas pessoas, em especial as mais idosas, perceberão que a imagem treme (pois a imagem anterior some da retina antes que a nova apareça). Em vez de aumentar a taxa de quadros, o que exigiria utilizar ainda mais a já escassa largura de banda, uma outra estratégia foi

adotada. Em lugar de exibir as linhas de varredura ordenadamente, primeiro todas as linhas ímpares são exibidas e depois todas as linhas pares. Cada uma dessas metades de quadro é chamada **campo.** As experiências comprovaram que, embora as pessoas percebam a oscilação da imagem a uma taxa de 25 quadros por segundo, elas não percebem o fenômeno com uma taxa de 50 campos por segundo. Essa técnica é chamada **entrelaçamento.** A televisão ou o vídeo não entrelaçado denomina-se **progressivo.** Observe que os filmes são exibidos a 24 quadros por segundo, mas cada quadro fica totalmente visível durante 1/24 segundo.

O vídeo colorido segue o mesmo padrão de varredura do vídeo monocromático (preto e branco), exceto pelo fato de, em vez de exibir a imagem com um feixe em movimento, são utilizados três feixes que se movem em uníssono. Cada feixe é usado para uma das três cores primárias: vermelho (red), verde (green) e azul (blue) — o padrão RGB. Essa técnica funciona porque é possível criar qualquer cor a partir da superposição linear de vermelho, verde e azul, com as intensidades apropriadas. Entretanto, para transmissão em um único canal, os três sinais de cores devem ser combinados em um único sinal **composto.**

Quando a televisão em cores foi criada, vários métodos para exibição de cores eram tecnicamente possíveis, e os diferentes países fizeram escolhas distintas, criando sistemas que até hoje ainda são incompatíveis. (Observe que essas escolhas nada têm a ver com VHS, Betamax e P2000, que são métodos de gravação.) Em todos os países, havia uma exigência política segundo a qual os aparelhos de televisão em branco e preto existentes deveriam ter a possibilidade de receber todos os programas transmitidos em cores. Portanto, o esquema mais simples, apenas codificar os sinais RGB separadamente, não era aceitável. Além disso, o RGB não é o esquema mais eficiente.

O primeiro sistema de cores foi padronizado nos Estados Unidos pelo **National**

Television Standards Committee, que emprestou seu acrônimo ao padrão: **NTSC.** A televisão em cores foi introduzida na Europa vários anos mais tarde, quando a tecnologia já tinha progredido substancialmente, dando origem a sistemas com maior imunidade a ruídos e cores melhores. Esses sistemas são chamados **SECAM (SEquentiel Couleur Avec Memoire)**, usado na França e na Europa Oriental, e **PAL (Phase Alternating Line),** usado no restante da Europa. A diferença na qualidade das cores entre o NTSC e o PAL/SECAM deu origem a uma piada no setor segundo a qual NTSC na verdade significa Never Twice the Same Color (nunca a mesma cor duas vezes).

Para permitir que transmissões a cores fossem vistas em aparelhos em branco e preto, os três sistemas combinam linearmente os sinais RGB com um sinal de **luminância** (brilho) e com dois sinais de **crominância** (cores), embora utilizem diferentes coeficientes para a criação desses sinais a partir dos sinais RGB. É interessante observar que o olho é muito mais sensível ao sinal de luminância que aos sinais de crominância e, portanto, esses últimos não precisam ser transmitidos com a mesma precisão. Em conseqüência disso, o sinal de luminância pode ser transmitido na mesma freqüência que o antigo sinal em branco e preto, de modo que esse sinal seja recebido por aparelhos de televisão em branco e preto. Os dois sinais de crominância são difundidos por bandas estreitas e em freqüências mais altas. Alguns aparelhos de TV têm botões de brilho, matiz e saturação (ou brilho, matiz e cor) para possibilitar o controle desses três sinais separadamente. Para entendermos como funciona a compactação de vídeo, é necessário entendermos os conceitos de luminância e crominância.

Nos últimos anos, temos visto um interesse considerável pela **HDTV (High Definition TeleVision)**, que produz imagens mais nítidas, pois praticamente duplica o número de linhas de varredura. Os Estados Unidos, a Europa e o Japão

já desenvolveram sistemas HDTV, todos diferentes e incompatíveis entre si. Os princípios básicos da HDTV, em termos de varredura, luminância, crominância etc. são semelhantes aos sistemas existentes. Entretanto, os três formatos têm uma relação entre eixos de 16:9 em vez de 4:3 para se aproximarem mais do formato usado no cinema (onde os filmes são gravados em filmes de 35 mm que têm uma relação entre eixos de 3:2).

[T4] Sistemas digitais

A mais simples representação de um sinal de vídeo digital é uma seqüência de quadros, cada um formado por uma grade retangular de elementos de imagens ou **pixels**. Cada pixel pode ser um único bit, cujo objetivo é representar um ponto preto ou branco. A qualidade desse sistema é a mesma que se obtém ao enviar uma fotografia colorida por fax — horrível. (Faça essa experiência ou então tire uma fotocópia de uma fotografia colorida em uma máquina que não efetue a conversão em mapa de bits.)

O próximo passo é usar 8 bits por pixel para representar 256 tons de cinza. Esse esquema gera imagens de alta qualidade em preto e branco. Para imagens coloridas, os melhores sistemas utilizam 8 bits para cada uma das cores RGB, embora quase todos os sistemas os misturem em um vídeo composto para transmissão. Ainda que o uso de 24 bits por pixel limite o número de cores a aproximadamente 16 milhões, o olho humano não consegue nem mesmo distinguir tantas cores, imagine mais. As imagens coloridas digitais são produzidas usando-se três feixes de varredura, um para cada cor. A geometria é a mesma do sistema analógico mostrado na Figura 7.70; a diferença é que as linhas de varredura contínuas são substituídas por linhas de pixels descontínuas. Para produzir movimentos suaves, o sinal de vídeo digital, assim como o sinal analógico, deve apresentar pelo menos 25 quadros por segundo. Entretanto,

como os monitores de boa qualidade quase sempre repetem a varredura da tela a partir de imagens armazenadas na memória a uma freqüência 75 vezes por segundo ou mais, o entrelaçamento não é necessário e, portanto, normalmente não é usado. Pintar (ou seja, redesenhar) o mesmo quadro três vezes consecutivas é suficiente para eliminar a oscilação.

Em outras palavras, a suavidade de movimento é determinada pelo número de imagens *diferentes* por segundo, enquanto a oscilação é determinada pelo número de vezes que a tela é pintada por segundo. Esses dois parâmetros são diferentes. Uma imagem fixa pintada a uma taxa de 20 quadros por segundo não exibirá movimentos bruscos mas oscilará, porque um quadro desaparecerá da retina antes do próximo ser mostrado. Um filme com 20 quadros diferentes por segundo, cada um dos quais pintado quatro vezes consecutivas, não oscilará, mas os movimentos parecerão bruscos.

A importância desses dois parâmetros fica clara quando consideramos a largura de banda necessária para a transmissão de um sinal de vídeo digital por uma rede. Todos os monitores atuais utilizam uma relação entre eixos de 4:3 que, portanto, podem usar tubos de imagem de baixo preço produzidos em massa e projetados para o mercado consumidor de televisores. As configurações comuns são 1024 × 768, 1280 × 960 1600 × 1200. Até mesmo a menor dessas configurações, com 24 bits por pixel e 25 quadros por segundo, deve ser gerada a 472 Mbps. Seria necessário uma portadora SONET OC-12 para administrar isso, e usar uma portadora SONET OC-12 em todas as residências não está nos planos de ninguém. Duplicar a taxa para evitar a oscilação é uma idéia ainda menos atraente. Uma solução melhor seria transmitir 25 quadros por segundo e fazer com que o computador armazenasse cada um deles e o pintasse duas vezes. A televisão por difusão não utiliza essa estratégia, pois os aparelhos de TV não têm memória e, mesmo que tivessem memória, os sinais analógicos não poderiam ser

armazenados na memória RAM sem antes serem convertidos em sinais digitais, o que exige o uso de mais componentes de hardware. Como conseqüência disso, o entrelaçamento é necessário para a televisão por difusão, mas não para o vídeo digital.

[T3] 7.4.7 Compactação de vídeo

Já deve estar claro que a transmissão de vídeo não compactado está absolutamente fora de questão. A única esperança é que uma compactação maciça seja possível. Felizmente, as pesquisas nas últimas décadas nos levaram a muitas técnicas e algoritmos de compactação que tornam a transmissão de vídeo possível. Nesta seção, vamos estudar alguns métodos para a compactação de dados de vídeo.

Todos os sistemas de compactação exigem dois algoritmos: um para a compactação dos dados na origem, e outro para a descompactação no destino. Na literatura especializada, esses algoritmos são chamados algoritmos de **codificação** e de **decodificação**, respectivamente. Também vamos usar essa terminologia aqui.

Esses algoritmos exibem certas assimetrias importantes que devemos entender. Primeiro, para muitas aplicações, um documento de multimídia, digamos um filme, só será codificado uma vez (ao ser armazenado em um servidor de multimídia), mas será decodificado milhares de vezes (quando for visto pelos usuários). Essa assimetria significa que é aceitável o algoritmo de codificação ser lento e exigir um hardware dispendioso, desde que o algoritmo de decodificação seja rápido e não precise de hardware muito caro. Afinal de contas, o operador de um servidor de multimídia pode estar disposto a alugar um supercomputador paralelo durante algumas semanas para codificar toda a sua biblioteca de vídeo. No entanto, exigir que os clientes aluguem um supercomputador por duas horas

para ver um vídeo talvez não seja uma grande idéia. Muitos sistemas de compactação práticos fazem o máximo para tornar a decodificação rápida e simples, mesmo que isso signifique que a codificação tenha de ser lenta e complicada.

Por outro lado, para recursos de multimídia de tempo real, como a videoconferência, a codificação lenta é inaceitável. A codificação deve ocorrer automaticamente em tempo real. Portanto, os recursos de multimídia em tempo real utilizam diferentes algoritmos ou parâmetros, em vez de armazenar imagens de vídeo em disco, freqüentemente com uma compactação bem menor.

Uma segunda assimetria é que o processo de codificação/decodificação não precisa ser recíproco. Ou seja, ao compactar um arquivo, transmiti-lo e depois descompactá-lo, um usuário espera ter de volta o arquivo original, exatamente como ele era antes, do primeiro ao último bit. Com os recursos de multimídia, esse requisito não existe. É aceitável que o sinal de vídeo seja ligeiramente diferente do original depois da codificação e da posterior decodificação. Quando a saída decodificada não é exatamente igual à entrada original, o sistema é dito **com perdas**. Se a entrada e a saída são idênticas, o sistema é **sem perdas**. Os sistemas com perdas são importantes porque aceitar a perda de um pequeno volume de informações pode oferecer uma grande vantagem em termos de taxa de compactação possível.

[T4] O padrão JPEG

O vídeo é apenas uma seqüência de imagens (somada ao som). Se pudéssemos encontrar um bom algoritmo para codificar uma única imagem, esse algoritmo poderia ser aplicado a cada imagem sucessiva para realizar a compactação de vídeo. Existem bons algoritmos para compactação de imagens estáticas, e portanto vamos iniciar nosso estudo da compactação de vídeo por esses

algoritmos. O padrão **JPEG (Joint Photographic Experts Group)** para compactação de imagens estáticas e de tom contínuo (por exemplo, fotografias) foi desenvolvido por especialistas em fotografia que trabalharam sob os esforços conjuntos da ITU, ISO e IEC, um outro grupo de padrões. Ele é importante para a multimídia porque, para uma primeira aproximação, o padrão multimídia para imagens em movimento, MPEG, é apenas a codificação JPEG separada de cada quadro, com alguns recursos extras para a compactação entre quadros e para a detecção de movimentos. O JPEG está definido no padrão internacional 10918.

O JPEG tem quatro modos e muitas opções. Ele se parece mais com uma lista de compras do que com um simples algoritmo. Porém, para nossos objetivos só o modo seqüencial com perdas é relevante, e esse é o que está ilustrado na Figura 7.71. Além disso, vamos nos concentrar no modo como o JPEG é usado normalmente para codificar imagens de vídeo RGB de 24 bits, e deixaremos de lado alguns detalhes secundários em favor da simplicidade.

[arte: ver original p. 697]

[Dísticos]

[1] Entrada

[2] Preparação de bloco

[3] Transformação discreta de cossenos

[4] Quantização

[5] Quantização diferencial

[6] Codificação run-length

[7] Codificação de saída estatística

[8] Saída

[F]Figura 7.71

[FL] A operação do JPEG em modo seqüencial com perdas

O primeiro passo para a codificação de uma imagem com JPEG é a preparação do bloco. Para sermos mais específicos, vamos supor que a entrada do JPEG é uma imagem RGB de 640 × 480 com 24 bits por pixel, como mostra a Figura 7.72(a). Como o uso da luminância e da crominância oferece uma compactação melhor, primeiro vamos calcular a luminância, *Y*, e as duas crominâncias, *I* e *Q* (para o NTSC), de acordo com as seguintes fórmulas:

Y = 0,30R + 0,59G + 0,11B

I = 0,60R – 0,28G – 0,32B

Q = 0,21R – 0,52G + 0,31B

Para o sistema PAL, as crominâncias são chamadas *U* e *V* e os coeficientes são diferentes, mas a idéia é a mesma. O SECAM é diferente tanto do NTSC quanto do PAL.

São criadas matrizes separadas para *Y*, *I* e *Q*, cada uma com elementos na faixa de 0 a 255. Em seguida, são usados blocos quadrados de quatro pixels para a divisão proporcional das matrizes de *I* e *Q*, a fim de reduzi-las a 320 × 240. Há perdas nessa redução, que são praticamente imperceptíveis, pois o olho humano reage mais à luminância que à crominância. Contudo, essa redução compacta os dados em um fator de dois. Portanto, 128 é subtraído de cada elemento das três matrizes para inserir 0 no meio da faixa. Por último, cada matriz é dividida em blocos de 8 × 8. A matriz *Y* tem 4800 blocos; as outras duas têm 1200 blocos cada uma, como mostra a Figura 7.72(b).

[arte: ver original p. 698a]

[Dísticos]

[1] RGB

640

480

Pixel de 24 bits

(a)

[2] Y

640

Pixel de 8 bits

480 1 bloco

Bloco 4799

(b)

[3] I

320

240

240

Q

[F]Figura 7.72

[FL] (a) Dados RGB de entrada. (b) Depois da preparação do bloco

O segundo passo do JPEG é aplicar uma transformação discreta de cosseno, ou **DCT (Discrete Cosine Transformation)** a cada um dos 7200 blocos separadamente. A saída de cada DCT é uma matriz de 8 × 8 de coeficientes DCT. O elemento DCT (0,0) é o valor médio do bloco. Os outros elementos informam quanta potência espectral está presente em cada freqüência espacial. Teoricamente, uma transformação DCT não tem perdas mas, na prática, o uso de números de ponto flutuante e funções transcendentais sempre acarreta alguns erros de arredondamento, que resultam em uma pequena perda de informações. Normalmente, esses elementos desaparecem com rapidez com o aumento da distância em relação à origem (0,0), como sugere a Figura 7.73.

[arte: ver original p. 698b]

[Dísticos]

[1] Amplitude de Y/I/Q

y

x

[2] DCT

Fy

Fx

[F]Figura 7.73

[FL] (a) Um bloco da matriz *Y*. (b) Os coeficientes DCT

Depois que a DCT se completa, o JPEG passa para a etapa 3, chamada **quantização**, na qual os coeficientes DCT menos importantes são descartados. Essa transformação (com perdas) é realizada através da divisão de cada coeficiente da matriz DCT 8 × 8 por um peso obtido em uma tabela. Se todos os pesos forem iguais a 1, a transformação nada fará. Entretanto, se os pesos aumentarem muito a partir da origem, freqüências espaciais mais altas serão abandonadas rapidamente.

Um exemplo dessa etapa é mostrado na Figura 7.74. Nela, vemos a matriz DCT inicial, a tabela de quantização e o resultado obtido ao se dividir cada elemento DCT pelo elemento correspondente na tabela de quantização. Os valores na tabela de quantização não fazem parte do padrão JPEG. Cada aplicação deve fornecer seus próprios valores, garantindo assim o controle da negociação entre perdas e compactação.

[arte: ver original p. 699]

[Dísticos]

[1] Coeficientes DCT Tabela de quantização Coeficientes quantizados

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem

seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 7.74

[FL] Cálculo dos coeficientes DCT quantizados

A etapa 4 reduz o valor (0,0) de cada bloco (o valor do canto superior esquerdo) substituindo-o pela diferença entre ele e o elemento correspondente no bloco anterior. Como são as médias de seus respectivos blocos, esses elementos devem mudar lentamente; portanto, ao retiramos os valores diferenciais, devemos reduzir a maioria desses elementos a valores pequenos. Nenhum diferencial é calculado a partir dos outros valores. Os valores (0,0) são conhecidos como componentes DC, enquanto os outros valores são os componentes AC.

A etapa 5 torna linear os 64 elementos e aplica a codificação run-length à lista. A varredura do bloco da esquerda para a direita e depois de cima para baixo não fará com que os zeros sejam concentrados e, portanto, é usado um padrão de varredura em ziguezague, como mostra a Figura 7.75. Nesse exemplo, o padrão em ziguezague acaba produzindo 38 zeros consecutivos no fim da matriz. Esse string pode ser reduzido a uma simples contagem informando que há 38 zeros, uma técnica conhecida como **codificação run-length**.

Agora, temos uma lista de números que representam a imagem (em espaço de transformação). A etapa 6 é a codificação de Huffman dos números para armazenamento ou transmissão, atribuindo códigos mais curtos aos números comuns e mais longos aos números pouco comuns.

O JPEG talvez pareça complicado, mas isso acontece porque de fato ele *é* complicado. Mesmo assim, como freqüentemente produz uma compactação de 20:1 ou mais, ele é amplamente utilizado. A decodificação de uma imagem JPEG exige a execução do algoritmo de trás para frente. O JPEG é aproximadamente simétrico: a decodificação demora tanto quanto a codificação. Essa propriedade

não é verdadeira para todos os algoritmos de compactação, como veremos agora.

[arte: ver original p. 700]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 7.75

[FL] A ordem na qual os valores quantizados são transmitidos

[T4] O padrão MPEG

Finalmente, chegamos ao centro da questão: os padrões **MPEG (Motion Picture Experts Group).** Esses padrões são os principais algoritmos usados para a compactação de vídeos e têm sido adotados como padrões internacionais desde 1993. Como os filmes contêm tanto sons quanto imagens, o MPEG pode compactar igualmente áudio e vídeo. Já examinamos a compactação de áudio e a compactação de imagens estáticas; portanto, vamos estudar agora a compactação de vídeo.

O primeiro padrão a ser finalizado foi o MPEG-1 (padrão internacional 11172). Seu objetivo era produzir uma saída com qualidade de gravador de vídeo (352 × 240 para NTSC) usando uma taxa de bits de 1,2 Mbps. Uma imagem de 352 × 240 com 24 bits/pixel e 25 quadros/s exige 50,7 Mbps; então, reduzir essa taxa para 1,2 Mbps não é uma tarefa muito fácil. É necessário um fator de compactação igual a 40. O MPEG-1 pode ser transmitido por linhas de transmissão de par trançado por distâncias modestas. O MPEG-1 também é usado para armazenar filmes em CD-ROM.

O próximo padrão da família MPEG foi o MPEG-2 (padrão internacional 13818), projetado originalmente para compactar vídeos com qualidade de difusão em uma faixa de 4 a 6 Mbps; portanto, ele poderia caber em um canal de difusão

NTSC ou PAL. Mais tarde, o MPEG-2 foi expandido para aceitar resoluções mais altas, incluindo a tecnologia HDTV. Ele é muito comum agora, pois forma a base para o DVD e para a televisão digital por satélite.

Os princípios básicos dos MPEG-1 e MPEG-2 são semelhantes, mas os detalhes são diferentes. À primeira vista, o MPEG-2 é um super conjunto do MPEG-1, com mais recursos, formatos de quadros e opções de codificação. Descreveremos primeiro o MPEG-1 e depois o MPEG-2.

O MPEG-1 tem três partes: áudio, vídeo e sistema, que integra as outras duas partes, como mostra a Figura 7.76. Os codificadores de áudio e vídeo funcionam de maneira independente, o que levanta a questão de como os dois fluxos se sincronizam no receptor. Esse problema é resolvido com um clock do sistema de 90 kHz cuja saída é o valor da hora atual para ambos os codificadores. Esses valores têm 33 bits, a fim de permitir que os filmes sejam executados durante 24 horas sem se repetirem. Esses timbres de hora são incluídos na saída codificada e propagados até o receptor, que então pode sincronizar os fluxos de áudio e vídeo.

[arte: ver original p. 701]

[Dísticos]

[1] Sinal de áudio   Codificador de áudio

[2] Clock

[3] Sinal de vídeo   Codificador de vídeo

[4] Multiplexador do sistema   Saída MPEG-1

[F]Figura 7.76

[FL] Sincronização dos fluxos de áudio e vídeo no MPEG-1

Agora vamos considerar a compactação de vídeo do MPEG-1. Nos filmes, há dois tipos de redundância: espacial e temporal. O MPEG-1 utiliza as duas. A

redundância espacial pode ser utilizada pela simples codificação separada de cada quadro com o JPEG. Essa estratégia é utilizada às vezes, em especial quando há necessidade de acesso aleatório a cada quadro, como ocorre na edição de produções de vídeo. Dessa forma, é possível obter uma largura de banda compactada de 8 a 10 Mbps.

Uma compactação adicional pode ser obtida aproveitando-se o fato de que quadros consecutivos freqüentemente são quase idênticos. O efeito é menor do que parece a princípio, pois muitos cineastas efetuam cortes entre cenas a cada 3 ou 4 segundos (faça a cronometragem de um filme e conte as cenas). Contudo, até mesmo uma série de 75 quadros muito semelhantes oferece uma boa redução potencial em comparação com a simples codificação de cada quadro separado com JPEG.

Para cenas em que a câmera e o fundo da cena permanecem estáticos e um ou dois atores se movimentam lentamente, quase todos os pixels serão idênticos de um quadro para outro. Nesse caso, subtrair cada quadro de seu antecessor e efetuar a compactação JPEG sobre a diferença funcionaria muito bem. Entretanto, para cenas nas quais a câmera faz uma tomada panorâmica ou muda rapidamente de plano, essa técnica é insuficiente. É necessária alguma forma de compensar esse movimento. Isso é exatamente o que o MPEG faz; essa também é a diferença mais importante entre o MPEG e o JPEG.

A saída do MPEG-1 consiste em quatro tipos de quadros:

1. Quadros I (Intracoded): Imagens estáticas, independentes e codificadas com o JPEG.
2. Quadros P (Predictive): Diferença bloco a bloco em relação ao último quadro.
3. Quadros B (Birectional): Diferenças entre o último quadro e o quadro seguinte.
4. Quadros D (DC-coded): Médias de blocos usadas para o avanço rápido.

Os quadros I são apenas imagens estáticas codificadas com uma variante do JPEG,

e que também utilizam luminância de resolução completa e crominância de meia resolução ao longo de cada eixo. É necessário que os quadros I apareçam periodicamente no fluxo de saída por três motivos. Primeiro, o MPEG-1 pode ser usado para uma transmissão por multidifusão, com os espectadores ajustando a transmissão à vontade. Se todos os quadros dependessem de seus antecessores para voltar ao primeiro quadro, quem perdesse o primeiro quadro nunca poderia decodificar qualquer quadro subseqüente. Em segundo lugar, se qualquer quadro fosse recebido com erro, a decodificação adicional não seria mais possível. Em terceiro lugar, sem os quadros I, enquanto estivesse executando um avanço ou retrocesso rápido, o decodificador teria de calcular todos os quadros exibidos, de modo a saber o valor total do quadro em que parou. Por essas razões, os quadros I são inseridos na saída uma ou duas vezes por segundo.

Ao contrário, os quadros P codificam as diferenças entre os quadros. Eles se baseiam na idéia de **macroblocos**, que cobrem 16 × 16 pixels em espaço de luminância e 8 × 8 pixels em espaço de crominância. Um macrobloco é codificado pesquisando-se o quadro anterior em busca dele ou de algo apenas um pouco diferente dele.

Um exemplo de onde os quadros P seriam úteis é mostrado na Figura 7.77. Nela, vemos três quadros consecutivos que têm o mesmo fundo de cena, mas são diferentes na posição de uma pessoa. Os macroblocos contendo o fundo de cena serão exatamente iguais, mas os macroblocos contendo a pessoa terão um deslocamento na posição medido por algum valor desconhecido e terão de ser localizados.

[arte: ver original p. 702]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 7.77

[FL] Três quadros consecutivos

O padrão MPEG-1 não especifica como pesquisar, até onde pesquisar, nem determina a proximidade entre os valores para que a comparação seja considerada válida. Isso depende de cada implementação. Por exemplo, uma implementação pode pesquisar um macrobloco na posição atual do quadro anterior e em todas as outras posições deslocadas $\pm \Delta x$ na direção x e $\pm \Delta y$ na direção y. Para cada posição, o número de correspondências na matriz de luminância pode ser calculado. A posição com o maior número de pontos deve ser declarada vencedora, desde que permaneça acima do limiar previamente determinado. Caso contrário, o macrobloco seria declarado ausente. Algoritmos muito mais sofisticados também são possíveis, é claro.

Se um macrobloco for encontrado, ele será codificado verificando-se a diferença em relação a seu valor no quadro anterior (para a luminância e ambas as crominâncias). A essas matrizes diferença é aplicada a transformação discreta de cosseno, quantização, codificação run-length e codificação de Huffman, exatamente como acontece com o JPEG. O valor para o macrobloco no fluxo de saída é então o vetor de movimento (ou seja, que distância o macrobloco se deslocou a partir de sua posição anterior em cada direção), seguido pela lista de números da codificação de Huffman. Se o macrobloco não for localizado no quadro anterior, o valor atual será codificado com JPEG, assim como em um quadro I.

É óbvio que esse algoritmo é altamente assimétrico. Uma implementação é livre para experimentar todas as posições plausíveis no quadro anterior, em uma tentativa desesperada de localizar todos os últimos macroblocos, não importando para onde ele se deslocou. Essa estratégia minimizará o fluxo de codificação

MPEG-1 a expensas de uma codificação muito lenta. Essa solução deve ser boa para uma codificação que só ocorra uma vez, como a codificação de uma filmoteca, mas seria terrível para videoconferências em tempo real.

Da mesma forma, cada implementação é livre para decidir o que constitui um macrobloco "encontrado". Essa liberdade permite aos implementadores competirem pela qualidade e pela velocidade de seus algoritmos, mas sempre produz algoritmos que obedecem ao MPEG-1. Independente do algoritmo de pesquisa utilizado, a saída final é a codificação JPEG do macrobloco atual, ou a codificação JPEG da diferença entre o macrobloco atual e um macrobloco contido no quadro anterior que esteja a uma determinada distância do atual.

Até agora, a decodificação MPEG-1 tem se mostrado simples. Decodificar quadros 1 é o mesmo que decodificar imagens JPEG. Decodificar quadros P exige que o decodificador armazene o quadro anterior em um buffer e depois construa o novo quadro em um segundo buffer baseado em macroblocos completamente codificados e macroblocos contendo diferenças em relação ao quadro anterior. O novo quadro é montado um macrobloco por vez.

Os quadros B são semelhantes aos quadros P, exceto pelo fato de permitirem que o macrobloco de referência esteja em um quadro anterior ou em um quadro seguinte. Essa liberdade adicional acarreta uma melhoria na compensação de movimento, e também é útil quando objetos passam pela frente ou por trás de outros objetos. Para realizar a codificação de quadros, o codificador precisa manter três quadros decodificados na memória ao mesmo tempo: o quadro anterior, o atual e o próximo. Ainda que os quadros B ofereçam a melhor compactação, nem todas as implementações os aceitam.

Os quadros D só são usados para possibilitar a exibição de uma imagem de baixa resolução quando um avanço ou um retrocesso rápido está sendo realizado. A decodificação MPEG-1 normal em tempo real é bastante difícil. Esperar que o

decodificador a realize enquanto o vídeo funciona em uma velocidade igual a dez vezes a velocidade normal é querer demais. Em vez disso, os quadros D são usados para produzir imagens de baixa resolução. Cada entrada de quadro D é apenas o valor médio de um bloco, sem maiores codificações, o que facilita a exibição em tempo real. Esse recurso é importante, pois permite às pessoas examinar um vídeo em alta velocidade na busca de uma determinada cena.

Após terminar nosso estudo do MPEG-1, passaremos ao MPEG-2. A codificação MPEG-2 é fundamentalmente semelhante à codificação MPEG-1, com quadros I, P e B. Entretanto, os quadros D não são aceitos. Além disso, a transformação discreta de cosseno utiliza um bloco de 10 × 10 em lugar de um bloco de 8 × 8, a fim oferecer um número de coeficientes 50% maior, e portanto melhor qualidade. Como o MPEG-2 se destina à televisão por difusão, assim como a aplicações em CD-ROM, ele aceita tanto imagens progressivas quanto entrelaçadas, enquanto o MPEG-1 aceita somente imagens progressivas. Os dois padrões também são diferentes em outros pequenos detalhes.

Em vez de aceitar somente um nível de resolução, o MPEG-2 aceita quatro níveis: baixa (352 × 240), principal (720 × 480), alta-1440 (1440 × 1152) e alta (1920 × 1080). A resolução baixa se destina aos aparelhos de videocassete e a manter compatibilidade retroativa com o MPEG-1. A resolução principal se destina à difusão NTSC, e as outras duas são destinadas à HDTV. Para obter saída de alta qualidade, o MPEG funciona normalmente em 4 a 8 Mbps.

[T3] 7.4.8 Vídeo por demanda

O vídeo por demanda às vezes é comparado a uma loja eletrônica para locação de vídeos. O usuário (cliente) seleciona qualquer um dos vídeos disponíveis e o leva para assistir em casa. A única diferença é que com o vídeo por demanda a seleção é feita em casa, com o controle remoto do aparelho de TV, e o vídeo é exibido

imediatamente. Não é necessário ir até a loja. Não é preciso dizer que a implementação do vídeo por demanda é muito mais complicada que descrever seu funcionamento. Nesta seção, apresentaremos uma visão geral das idéias básicas e de sua implementação.

O vídeo por demanda é realmente semelhante à locação de um vídeo, ou se parece mais com a escolha de um filme a partir de um sistema de TV a cabo com 500 canais? A resposta tem importantes implicações técnicas. Em particular, os usuários de locadoras de vídeo estão acostumados à idéia de poderem interromper a exibição do vídeo, ir até a cozinha ou ao banheiro e depois retomar de onde haviam parado. Os espectadores de televisão não esperam poder fazer uma pausa na exibição dos programas.

Se o vídeo por demanda pretende competir de maneira bem-sucedida com as locadoras de vídeo, talvez seja necessário permitir aos usuários interromper, recomeçar e retroceder os vídeos à vontade. Para proporcionar esses recursos aos usuários, os provedores de vídeo são praticamente forçados a transmitir uma cópia separada do vídeo para cada um desses usuários.

Por outro lado, se o vídeo por demanda pretende ser apenas uma televisão mais avançada, talvez seja suficiente que o provedor de vídeo comece os vídeos mais solicitados a cada, digamos, 10 minutos, e apresente-os sem intervalo. Um usuário que desejar assistir a um desses vídeos talvez tenha de esperar 10 minutos para que ele comece. Embora a pausa/retomada não seja possível aqui, um espectador que retorne à sala depois de um pequeno intervalo pode mudar para outro canal onde o mesmo vídeo está sendo exibido com 10 minutos de atraso. Haverá alguma repetição, mas nada será perdido. Esse esquema é chamado **vídeo quase por demanda (near video on demand)** e oferece um custo potencial muito menor, pois a mesma imagem do servidor de vídeo pode ir para vários usuários ao mesmo tempo. A diferença entre o vídeo por demanda e o

método vídeo quase por demanda é semelhante à diferença entre dirigir seu próprio carro e pegar um ônibus.

Assistir a filmes no esquema de vídeo (quase) por demanda é um dos inúmeros serviços novos que poderão ser oferecidos, uma vez que as redes de banda larga estejam disponíveis. O modelo genérico usado por muitas pessoas é ilustrado na Figura 7.78. Na figura, vemos uma rede de backbone geograficamente distribuída (nacional ou internacional) de alta largura de banda no centro do sistema. Conectadas a ela estão milhares de redes de distribuição locais, tais como TV a cabo ou sistemas de distribuição da companhia telefônica. Os sistemas de distribuição locais chegam até as casas das pessoas, onde terminam em set-top boxes — na verdade, PCs avançados e especializados.

[arte: ver original p. 705]

[Dísticos]

[1] Servidor de vídeo

[2] Servidor de áudio

[3] Fibra

[4] Rede de backbone ATM ou SONET

[5] Switch

[6] Servidor local de spool

[7] Rede de distribuição local

[8] Casa do cliente

[F]Figura 7.78

[FL] Visão geral de um sistema de vídeo por demanda

Ligados ao backbone por cabos de fibra óptica de alta largura de banda estão milhares de provedores de informações. Alguns deles oferecerão vídeos pay-per-view, ou CDs de áudio pay-per-hear, nos quais o interessado paga para ver e

para ouvir, respectivamente. Outros oferecerão serviços especializados como fazer compras em casa (com a possibilidade de girar uma lata de sopa e acionar o zoom para ler a lista dos ingredientes ou ver um clipe sobre como conduzir um cortador de grama movido a gasolina). Esportes, notícias, retransmissões de "I Love Lucy", acesso à WWW e inúmeras outras possibilidades sem dúvida estarão disponíveis em um futuro próximo.

Também estão incluídos no sistema servidores locais de spool, que permitem que os vídeos fiquem mais próximos dos usuários (com antecedência), a fim de poupar largura de banda durante os horários de pico. A forma como essas peças se encaixarão e a quem caberá cada responsabilidade são questões para vigorosos debates no setor. A seguir, vamos examinar o projeto dos principais componentes do sistema: os servidores de vídeo e a rede de distribuição.

[T4] Servidores de vídeo

Para ter um recurso de vídeo (quase) por demanda, precisamos de **servidores de vídeo** com capacidade para armazenar e enviar um grande número de filmes simultaneamente. O número total de filmes já feitos é estimado em 65.000 (Minoli, 1995). Quando compactado em MPEG-2, um filme normal ocupa quase 4 GB de espaço de armazenamento; portanto, 65.000 filmes ocupariam algo em torno de 260 terabytes. Se acrescentarmos a tudo isso todos os antigos programas de televisão, filmes de esportes, documentários, anúncios de produtos etc., ficará claro que temos um problema de armazenamento de proporções industriais em nossas mãos.

A forma mais econômica de armazenar grandes volumes de informações é a fita magnética. Essa estratégia não é nova, e provavelmente continuará por muito tempo ainda. Uma fita Ultrium pode armazenar 200 GB (50 filmes) a um custo de cerca de aproximadamente 1 a 2 dólares por filme. Já existem no mercado

grandes servidores mecânicos que guardam milhares de fitas e que têm um braço de robô para pegar uma fita e inseri-la na unidade correspondente. O problema com esses sistemas é o tempo de acesso (em especial para o quinquagésimo filme em uma fita), a taxa de transferência e o número limitado de unidades de fita (para transmitir *n* filmes ao mesmo tempo, seriam necessárias *n* unidades de fita).

Felizmente, a experiência das locadoras de vídeo, bibliotecas públicas e outras organizações afins mostra que nem todos os itens têm a mesma popularidade. Constatou-se que quando há *N* filmes disponíveis, a fração de todas as solicitações para o *k*-ésimo filme mais popular é de aproximadamente C/k. Aqui, C é calculado a fim de normalizar a soma para 1, a saber:

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \ldots + 1/N)$$

Portanto, o filme mais popular é sete vezes mais popular que o filme número sete. Esse resultado é conhecido como lei de Zipf (Zipf, 1949).

O fato de que alguns filmes são muito mais populares que outros sugere uma solução possível na forma de uma hierarquia de armazenamento, como mostra a Figura 7.79. Na figura, o desempenho aumenta à medida que subimos pela hierarquia.

[arte: ver original p. 706]

[Dísticos]

[1] RAM

[2] Disco magnético

[3] DVD

[4] Fita

[F]Figura 7.79

[FL] A hierarquia de armazenamento de um servidor de vídeo

Uma alternativa para as fitas é o armazenamento óptico. Os DVDs atuais têm uma capacidade de 4,7 GB, boa para um filme, mas a próxima geração conterá dois filmes. Embora os tempos de busca sejam maiores se comparados aos discos magnéticos (50 ms *versus* 5 ms), seu baixo custo e sua alta confiabilidade fazem das juke boxes ópticas contendo milhares de DVDs uma boa alternativa para as fitas, no caso dos filmes mais usados.

A seguir, vêm os discos magnéticos. Eles têm tempos de acesso curtos (5 ms), altas taxas de transferência (320 MB/s para o SCSI 320) e grandes capacidades (> 100 GB), o que os torna apropriados para armazenar filmes que estejam sendo realmente transmitidos (em oposição a armazená-los para o caso de alguém querer vê-los). Sua grande desvantagem é o alto custo para o armazenamento de filmes que raramente são solicitados.

No topo da pirâmide da Figura 7.79 está a memória RAM. A memória RAM é o meio de armazenamento mais rápido e também o mais caro. Quando o preço da memória RAM cair para 50 dólares por gigabyte, um filme de 4 GB gastará 200 dólares de memória RAM; portanto, ter 100 filmes em memória RAM custará 20.000 dólares, no caso da memória de 200 GB. Mesmo assim, para um servidor de vídeo transmitir 100 filmes, a simples manutenção de todos os filmes em memória RAM está começando a se tornar viável. Além disso, se o servidor de vídeo tiver 100 clientes, mas todos eles estiverem assistindo a apenas 20 filmes diferentes, a idéia começará não apenas a parecer viável, mas também um bom projeto.

Tendo em vista que um servidor de vídeo na realidade é apenas um dispositivo de E/S em tempo real mais sofisticado, ele precisa de uma arquitetura de hardware e de software diferente da utilizada em um PC ou em uma estação de trabalho UNIX. A arquitetura de hardware de um servidor de vídeo típico é ilustrada na Figura 7.80. O servidor tem uma ou mais CPUs de alto desempenho, cada uma

com alguma memória local, uma memória principal compartilhada, um volumoso cache de memória RAM para os filmes populares, uma variedade de dispositivos de armazenamento para conter os filmes e um hardware de rede, normalmente uma interface óptica para um backbone SONET ou ATM à velocidade OC-12 ou maior. Esses subsistemas são conectados por um barramento de velocidade extremamente alta (pelo menos 1 GB/s).

[arte: ver original p. 707]

[Dísticos]

[1] CPU RAM local

[2] CPU RAM local

[3] RAM principal

[4] Cache de filmes (RAM)

Barramento de alta velocidade

[5] Controlador de fita

Arquivo de armazenamento de fitas

[6] Controlador de disco óptico

Juke box óptica

[7] Controlador de disco magnético

RAID

[8] Interface de rede

Para backbone

[F]Figura 7.80

[FL] A arquitetura de hardware de um servidor de vídeo típico

Agora, vamos examinar rapidamente o software do servidor de vídeo. As CPUs são usadas para receber as solicitações dos usuários, localizar os filmes, movimentar dados entre dispositivos, fazer a cobrança e muitas outras funções.

Algumas delas não dependem de tempo, mas muitas outras dependem; portanto, algumas CPUs (senão todas) terão de executar um sistema operacional de tempo real, como microkernel de tempo real. Esses sistemas normalmente dividem o trabalho em pequenas tarefas, cada uma com um prazo conhecido. Em seguida, o programador de execução (scheduler) pode executar um algoritmo como o do prazo seguinte mais próximo ou o algoritmo de taxa monotônico (Liu e Layland, 1973).

O software da CPU também define a natureza da interface que o servidor apresenta aos clientes (servidores de spool e set top boxes). Há dois projetos populares. O primeiro é um sistema de arquivos tradicional, no qual os clientes podem abrir, ler, gravar e fechar arquivos. Em vez das complicações exibidas pela hierarquia de armazenamento e as considerações sobre o tempo real, esse servidor pode ter um sistema de arquivos baseado no sistema do UNIX.

O segundo tipo de interface se baseia no modelo de gravador de vídeo (videocassete). Os comandos para o servidor solicitam que ele abra, reproduza, faça pausas, efetue o avanço e o retrocesso rápido de arquivos. A diferença em relação ao modelo UNIX é que, uma vez executado um comando *PLAY*, o servidor simplesmente continua a transmitir dados a uma taxa constante, sem a necessidade de novos comandos.

O núcleo do software do servidor de vídeo é o software de gerenciamento de disco. Ele tem duas funções principais: colocar os filmes no disco magnético quando tiverem de ser retirados da unidade de armazenamento óptico ou de fita e tratar as solicitações de disco para os muitos fluxos de saída. A colocação dos filmes é muito importante, porque pode afetar bastante o desempenho.

Dois métodos possíveis para a organização do armazenamento em disco são o @@@disk farm (grupo de discos) e o array de discos. Com o **disk farm**, cada unidade de disco mantém alguns filmes completos. Por motivos de desempenho e

confiabilidade, cada filme deve estar presente em pelo menos duas unidades de disco e, às vezes, em mais. Outro método de organizar o armazenamento é o **array de discos** ou **RAID (Redundant Array of Inexpensive Disks)**, no qual cada filme é distribuído entre várias unidades de discos, por exemplo, o bloco 0 na unidade 0, o bloco 1 na unidade 1 e assim por diante, com o bloco $n - 1$ na unidade $n - 1$. Depois disso, o ciclo se repete, com o bloco n na unidade 0 e assim sucessivamente. Esse método de organização é chamada **divisão em faixas (striping)**.

Um array de discos dividido em faixas apresenta diversas vantagens sobre um disk farm. Primeiro, todas as n unidades de discos podem ser executadas em paralelo, aumentando o desempenho n vezes. Em segundo lugar, ele pode se tornar redundante, caso uma unidade de discos extra seja acrescentada a cada grupo de n, onde a unidade redundante contém o resultado do OR exclusivo bloco por bloco das outras unidades de discos, a fim de permitir uma recuperação de dados completa na eventualidade de ocorrer uma falha de unidade. Por último, o problema do balanceamento da carga é resolvido (não é preciso uma disposição manual para evitar que todos os filmes mais populares fiquem na mesma unidade de disco). Por outro lado, a organização por array de discos é mais complicada que a organização por disk farm, e é altamente sensível a diversas falhas. Ela também não é apropriada para operações de videocassete, como rebobinamento ou avanço rápido de um filme.

A outra tarefa do software de disco é dar assistência a todos os fluxos de saída de tempo real e atender a suas restrições de tempo. Somente há alguns anos, isso exigia algoritmos complexos de programação de discos mas, com a queda atual dos preços de memória, está começando a se tornar possível uma abordagem muito mais simples. Para cada fluxo servido, é mantido em RAM um buffer de, digamos, 10 segundos de vídeo (5 MB). Ele é preenchido por um

processo de disco e esvaziado por um processo de rede. Com 500 MB de memória RAM, 100 fluxos podem ser alimentados diretamente a partir da RAM. É claro que o subsistema de disco deve ter uma taxa de dados sustentada de 50 MB/s para manter os buffers cheios, mas um RAID construído a partir de discos SCSI de alta tecnologia pode manipular com facilidade esse requisito.

[T4] A rede de distribuição

A rede de distribuição é o conjunto de switches e linhas entre a origem e o destino. Como vimos na Figura 7.78, ela consiste em um backbone conectado a uma rede de distribuição local. Em geral, o backbone é comutado, enquanto a rede de distribuição local não é.

O principal requisito imposto ao backbone é alta largura de banda. A baixa flutuação também costuma ser uma exigência mas, como até mesmo o menor PC de hoje é capaz de armazenar no buffer 10 segundos de vídeo MPEG-2 de alta qualidade, a baixa flutuação não é mais um requisito.

A distribuição local é caótica, com diferentes empresas experimentando diferentes redes em diferentes regiões. Companhias telefônicas, companhias de TV a cabo e novos participantes como as empresas de energia, estão todas convencidas de que a primeira a chegar será a grande vencedora; portanto, estamos vendo agora uma grande proliferação de tecnologias. No Japão, algumas empresas de esgotos estão no ramo da Internet, afirmando que têm o maior canal de todos para todas as residências (essas empresas fazem um cabo de fibra óptica passar pela tubulação de esgotos, mas precisam ter muito cuidado com o local exato onde ele emerge). Os quatro maiores esquemas de distribuição local para vídeo por demanda são representados pelos acrônimos: ADSL, FTTC, FTTH e HFC. Agora, vamos explicá-los, um de cada vez.

A **ADSL** foi a primeira participação da indústria telefônica no processo de

distribuição local. Estudamos a ADSL no Capítulo 2 e não repetiremos aqui esse assunto. A idéia é que praticamente todos os lares nos Estados Unidos, na Europa e no Japão já têm um par de fios de cobre trançados que chegam até eles (para o serviço de telefones analógicos). Se esses cabos pudessem ser usados para vídeo por demanda, as empresas telefônicas poderiam sair lucrando.

O problema é que esses cabos não aceitam nem mesmo o MPEG-1 em sua extensão típica de 10 km, muito menos o MPEG-2. O vídeo de alta resolução, em cores e de movimentação total necessita de 4 a 8 Mbps, dependendo da qualidade desejada. A ADSL não é realmente rápida o bastante, exceto em loops locais muito curtos.

O segundo projeto para companhias telefônicas é o **FTTC (Fiber To The Curb).** No FTTC, a companhia telefônica estende cabos de fibra óptica desde a estação final até cada bairro residencial, terminando em um dispositivo chamado **ONU (Optical Network Unit).** Cada ONU pode receber 16 loops locais de cobre. Atualmente, esses loops são tão curtos que é possível utilizar T1 ou T2 full duplex sobre eles, permitindo o transporte de filmes MPEG-1 e MPEG-2, respectivamente. Além disso, hoje em dia é possível ter recursos de videoconferência para pessoas que trabalham em casa ou para microempresas, pois o FTTC é simétrico.

A terceira solução para companhias telefônicas é estender cabos de fibra óptica até cada residência. Esse sistema é chamado **FTTH (Fiber To The Home).** Nesse esquema, qualquer um pode ter uma portadora OC-1, OC-3 ou até uma portadora mais alta, se for necessário. O FTTH é muito caro e não se consolidará nos próximos anos, mas sem dúvida abrirá uma grande variedade de novas possibilidades quando finalmente se consolidar. Na Figura 7.63, vimos que qualquer pessoa podia operar sua própria estação de rádio. O que você acha de cada membro da família operar sua própria estação de TV pessoal? A ADSL, o FTTC e o FTTH são redes de distribuição local ponto a ponto, o que não é

surpresa, considerando-se a organização atual do sistema telefônico.

Uma tecnologia bastante diferente é **HFC (Hybrid Fiber Coax)**, a solução preferida e que está sendo atualmente instalada por provedores de TV a cabo. O HFC é ilustrado na Figura 2.47(a). A história é mais ou menos esta: os cabos coaxiais atuais de 300 a 450 MHz estão sendo substituídos por cabos coaxiais de 750 MHz, aumentando a capacidade de 50 a75 canais de 6 MHz para 125 canais de 6 MHz. Setenta e cinco dos 125 canais serão usados para a transmissão de TV analógica.

Cada um dos 50 canais novos será modulado com QAM-256, que oferece cerca de 40 Mbps por canal, proporcionando mais 2 Gbps de largura de banda. Os head-ends serão deslocados mais para o interior dos bairros, de modo que cada cabo passe por somente 500 casas. Uma simples divisão mostra que cada casa poderá receber um canal dedicado de 4 Mbps, que pode manipular um filme MPEG-2.

Embora isso pareça maravilhoso, o HFC exige que todos os provedores de TV a cabo substituam os cabos existentes por cabos coaxiais de 750 MHz, instalem novos head-ends e removam todos os amplificadores unidirecionais — resumindo, que substituam o sistema de TV a cabo inteiro. Conseqüentemente, a infra-estrutura necessária aqui é comparável à infra-estrutura de que as companhias telefônicas precisam para o FTTC. Em ambos os casos, o provedor de rede local tem de estender cabos de fibra óptica até os bairros residenciais. Mais uma vez, nos dois casos, o cabo de fibra óptica termina em um conversor optoelétrico. No FTTC, o segmento final é um loop local ponto a ponto que utiliza pares trançados. No HFC, o segmento final é um cabo coaxial compartilhado. Em termos técnicos, esses dois sistemas não são tão diferentes quanto seus respectivos proponentes anunciam com freqüência.

Contudo, existe uma diferença real que vale a pena apontar. O HFC utiliza um

meio compartilhado sem comutação e roteamento. Qualquer informação incluída no cabo pode ser obtida por qualquer assinante sem maiores dificuldades. O FTTC, que funciona de forma totalmente comutada, não tem essa propriedade. Como resultado, as operadoras de HFC querem que os servidores de vídeo enviem fluxos criptografados; dessa forma, os clientes que não tiverem pago por um filme não poderão vê-lo. As operadoras FTTC não têm uma preferência especial pela criptografia, porque ela aumenta a complexidade, diminui o desempenho e não oferece qualquer segurança adicional a seu sistema. Do ponto de vista de uma companhia que opera um servidor de vídeo, é ou não uma boa idéia usar a criptografia? Um servidor operado por uma companhia telefônica ou por uma de suas subsidiárias ou parceiras poderia decidir não criptografar intencionalmente seus vídeos, alegando que o motivo é a eficiência mas, na verdade, sua intenção é causar prejuízos econômicos a seus concorrentes HFC.
Com todas essas redes de distribuição local, é provável que cada bairro seja equipado no futuro com um ou mais servidores de spool que, na realidade, são apenas versões menores dos servidores de vídeo que discutimos antes. A grande vantagem desses servidores locais é que eles retiram uma parte da carga do backbone.
Esses servidores spool podem ser pré-carregados com filmes por meio de reservas. Se as pessoas informarem ao provedor com bastante antecedência que filmes desejam assistir, eles poderão ser transferidos por download para o servidor local durante os horários de pouco movimento. Essa observação é capaz de levar as operadoras de rede a induzir os executivos das empresas aéreas a modificarem seus preços. É possível imaginar tarifas com um desconto de 27%, nas quais os filmes são solicitados com 24 a 72 horas de antecedência, para serem assistidos em uma terça-feira ou quinta-feira à noite, antes das 18 horas ou depois das 23 horas. Os filmes solicitados no primeiro domingo do mês antes

das 8 horas da manhã, para serem assistidos em uma tarde de quarta-feira cuja data for um número primo, ganharão um desconto de 43% e assim por diante.

[T3] 7.4.9 MBone — Multicast Backbone

Enquanto essas indústrias estão fazendo planos audaciosos — e com intensa publicidade — para o futuro vídeo digital (inter)nacional por demanda, a comunidade da Internet vem implementado em silêncio seu próprio sistema de multimídia digital, o **MBone (Multicast Backbone)**. Nesta seção, apresentaremos uma visão geral do que ele é e como funciona.

O MBone pode ser considerado a televisão na Internet. Ao contrário do vídeo por demanda, onde a ênfase está em fazer uma ligação e assistir a vídeos pré-compactados armazenados em um servidor, o MBone é usado para a transmissão de áudio e vídeo ao vivo em sinal digital em todo o mundo através da Internet. O MBone está em funcionamento desde o início de 1992. Muitas conferências científicas, em especial as reuniões da IETF, têm sido transmitidas, bem como eventos científicos dignos de divulgação, tais como o lançamento do ônibus espacial. Certa vez, um show dos Rolling Stones foi transmitido pelo MBone, como também partes do festival de cinema de Cannes. Se esses eventos podem ser classificados como científicos e dignos de divulgação é algo discutível.

Tecnicamente, o MBone é uma rede de overlay virtual montada sobre a Internet. Ele é formado por ilhas com capacidade de multidifusão conectadas por túneis, como mostra a Figura 7.81. Nessa figura, o MBone consiste em seis ilhas, de *A* a *F*, conectadas por sete túneis. Cada ilha (normalmente, uma LAN ou um grupo de LANs interconectadas) tem um hardware de multidifusão para comunicação com seus hosts. Os túneis propagam pacotes MBone entre as ilhas. No futuro, quando todos os roteadores forem capazes de lidar diretamente com o tráfego de multidifusão, essa superestrutura não será mais necessária, mas no momento ela

funciona.

Cada ilha contém um ou mais roteadores especiais chamados **mrouters (multicast routers)**. Na verdade, alguns deles são roteadores normais, mas a maioria é apenas uma estação de trabalho UNIX que executa um software especial no nível do usuário (mas como raiz). Os mrouters são conectados logicamente por túneis. Os pacotes MBone são encapsulados em pacotes IP e enviados como pacotes normais de unidifusão ao endereço IP do mrouter de destino.

[arte: ver original p. 712]

[Dísticos]

[1] Estação de trabalho

A

[2] Mrouter

B

Ilha de multidifusão

[3] C D

LAN

G

E F

[F]Figura 7.81

[FL] O MBone consiste em ilhas de multidifusão conectadas por túneis

Os túneis são configurados manualmente. Em geral, um túnel passa por um caminho no qual existe uma conexão física, mas isso não é obrigatório. Se, por acaso, o caminho físico sob o túnel tiver algum problema técnico, os mrouters que utilizam o túnel nem mesmo irão notá-lo, pois a Internet deverá reencaminhar automaticamente todo o tráfego IP entre eles por outras linhas. Quando uma nova ilha surge e deseja se juntar ao MBone, como é o caso de *G* na

Figura 7.81, seu administrador envia uma mensagem anunciando sua presença na lista de debate do Mbone. Em seguida, os administradores dos sites vizinhos entram em contato com ele para gerar os túneis. Às vezes, os túneis existentes são reordenados para se beneficiarem da nova ilha e otimizarem a topologia. Afinal de contas, os túneis não têm existência física. Eles são definidos por tabelas nos mrouters e podem ser acrescentados, excluídos ou deslocados pela simples alteração dessas tabela. Em geral, cada país no MBone tem um backbone, com ilhas regionais ligadas a ele. Normalmente, o MBone é configurado com um ou dois túneis que cruzam os oceanos Atlântico e Pacífico, fazendo do MBone um sistema uma escala global.

Portanto, em qualquer instante o MBone é formado por uma topologia específica que consiste em ilhas e túneis, independente do número de endereços de multidifusão em uso no momento e de quem está escutando ou inspecionando esses endereços. Essa situação é muito parecida com a de uma sub-rede (física) normal; portanto, é possível aplicar algoritmos de roteamento a ela. Conseqüentemente, em princípio o MBone utilizava um algoritmo de roteamento, o **DVMRP (Distance Vector Multicast Routing Protocol)**, baseado no algoritmo do vetor de distância de Bellman-Ford. Por exemplo, na Figura 7.81, a ilha *C* pode fazer o roteamento para *A* passando por *B* ou por *E* (ou até mesmo através de *D*). Para fazer uma escolha, a ilha *C* obtém os valores que esses nós fornecem como suas respectivas distâncias até *A* e, em seguida, acrescenta sua própria distância até eles. Dessa maneira, cada ilha determina a melhor rota para qualquer outra ilha. Entretanto, as rotas não são utilizadas exatamente assim, como veremos em breve.

Consideraremos agora a forma como a multidifusão realmente acontece. Para fazer a multidifusão de um programa de áudio ou vídeo, primeiro uma origem deve adquirir um endereço de multidifusão da classe D, que funciona como uma

freqüência de estação de rádio ou um número de canal. Os endereços da classe D são reservados por um programa que examina um banco de dados em busca de endereços de multidifusão livres. Muitas transmissões de multidifusão podem estar acontecendo ao mesmo tempo, e um host pode "sintonizar" aquela em que está interessado, escutando o endereço de multidifusão apropriado.

Cada mrouter envia, em intervalos regulares, um pacote de difusão IGMP limitado a sua ilha, perguntando quem está interessado em cada canal. Os hosts que desejarem (continuar a) receber um ou mais canais, retornarão um outro pacote IGMP como resposta. Essas respostas serão alternadas para evitar sobrecarregar a LAN local. Cada mrouter mantém uma tabela dos canais que ele tem de colocar em sua LAN para evitar desperdiçar largura de banda fazendo a multidifusão de canais que ninguém quer.

As transmissões de multidifusão se propagam pelo MBone da seguinte maneira: quando uma fonte de áudio ou de vídeo gera um novo pacote, ela o envia por multidifusão à sua ilha local, utilizando o recurso de multidifusão de hardware. Esse pacote é recebido pelo mrouter local, que envia cópias dele para todos os túneis aos quais está conectado.

Cada mrouter que receber o pacote verifica se ele veio pela melhor rota, ou seja, pela rota indicada por sua tabela para chegar até a origem (como se a origem fosse o destino). Se o pacote tiver chegado pela melhor rota, o mrouter enviará suas cópias para todos os outros túneis. Se tiver chegado por uma rota não ideal, o pacote será descartado. Portanto, por exemplo, na Figura 7.81, se as tabelas de *C* informarem que ele deve usar *B* para chegar até *A*, quando um pacote de multidifusão chegar a *C* vindo de *A* através de *B*, o pacote será copiado para *D* e *E*. Entretanto, quando um pacote de multidifusão de *A* chegar a *C* através de *E* (que não é o melhor caminho), ele simplesmente será descartado. Esse algoritmo é apenas o algoritmo de encaminhamento pelo caminho inverso que vimos no

Capítulo 5. Embora não seja perfeito, sua qualidade é boa e a implementação é muito simples.

Além de usar o encaminhamento pelo caminho inverso para evitar inundar a Internet, o campo *Time to live* do IP também é usado para limitar a abrangência da multidifusão. Cada pacote começa com um valor (determinado pela origem). A cada túnel é atribuído um peso. Um pacote só passa por um túnel se tiver peso suficiente; caso contrário, ele é descartado. Por exemplo, túneis transoceânicos normalmente são configurados com um peso 128; portanto, os pacotes podem ser limitados ao continente de origem, se receberem um *Time to live* inicial de 127 ou menos. Após a passagem por um túnel, o campo *Time to live* é decrementado de acordo com o peso do túnel.

Ainda que o algoritmo de roteamento do MBone funcione, muitas pesquisas têm se dedicado a seu aperfeiçoamento. Uma proposta mantém a idéia do roteamento do vetor de distância, mas torna o algoritmo hierárquico ao agrupar os sites do MBone em regiões e efetuar primeiro o roteamento para elas (Thyagarajan e Deering, 1995).

Outra proposta é usar uma forma modificada de roteamento por estado de enlace em vez de roteamento com vetor de distância. Em particular, um grupo de trabalho da IETF está muito ocupado modificando o OSPF para adequá-lo à multidifusão dentro de um único sistema autônomo. Esse OSPF de multidifusão é chamado **MOSPF** (Moy, 1994). As modificações têm o mapa completo criado pelo MOSPF para manter registros de ilhas de multidifusão e túneis, além das informações habituais sobre roteamento. Com conhecimento de toda a topologia, é mais rápido calcular o melhor caminho entre uma ilha e as outras utilizando os túneis. Por exemplo, pode-se usar o algoritmo de Dijkstra.

Uma segunda área de pesquisa é o roteamento entre sistemas autônomos. Para isso, um outro grupo de trabalho da IETF está desenvolvendo um algoritmo

chamado **PIM (Protocol Independent Multicast).** O PIM tem duas versões, dependendo do fato de as ilhas serem densas (ou seja, quase todos querem assistir) ou esparsas (quase ninguém quer assistir). Ambas as versões utilizam as tabelas de roteamento por unidifusão padrão, em vez de criarem uma topologia de overlay como fazem o DVMRP e o MOSPF.

No PIM-DM (modo denso), a idéia é suprimir os caminhos inúteis. Isso funciona da seguinte maneira: quando um pacote de multidifusão é recebido por um túnel "errado", um pacote de supressão (prune) é retornado pelo túnel, informando ao transmissor que ele deve parar de enviar pacotes da origem em questão. Quando chega através do túnel "certo", o pacote é copiado para todos os outros túneis que não tenham sido suprimidos anteriormente. Se todos os outros túneis tiverem sido suprimidos e não houver interesse nesse canal dentro da ilha local, o mrouter enviará uma mensagem de supressão pelo canal "certo". Dessa forma, a multidifusão se adapta automaticamente e só vai para onde é desejada.

O PIM-SM (modo esparso), descrito na RFC 2362, funciona de outra maneira. A idéia aqui é evitar a saturação da Internet, porque três pessoas em Berkeley desejam manter uma teleconferência através de um endereço da classe D. O PIM-SM funciona configurando pontos de encontro. Cada uma das origens em um grupo de multidifusão com PIM-SM envia seus pacotes aos pontos de encontro. Qualquer site interessado em se juntar ao grupo solicita a um dos pontos de encontro que estabeleça um túnel até ele. Dessa forma, todo o tráfego PIM-SM é transportado por unidifusão em vez de ser transportado por multidifusão. O PIM-SM está se tornando mais popular, e o MBone está migrando para seu uso. À medida que o PIM-SM se tornar mais usado, o MOSPF irá desaparecer gradualmente. Por outro lado, o próprio MBone parece um tanto estagnado, e talvez nunca venha a ser um sucesso.

Apesar de tudo isso, a multimídia em rede é um campo muito interessante e está

em rápida evolução, mesmo que o MBone não se torne um enorme sucesso.

Novas tecnologias e aplicações são anunciadas todos os dias. Cada vez mais, a multidifusão e a qualidade de serviço estão convergindo, conforme é descrito em (Striegel e Manimaram, 2002). Outro tópico interessante é a multidifusão sem fio (Gossain *et al.*, 2002). A área de multidifusão como um todo e tudo que está relacionado a ela deverão manter sua importância durante os próximos anos.

[T2] 7.5 Resumo

A atribuição de nomes na Internet utiliza um esquema hierárquico chamado sistema de nomes de domínios (DNS). No nível superior, encontram-se os domínios genéricos conhecidos, inclusive *com* e *edu*, bem como cerca de 200 domínios de países. O DNS é implementado sob a forma de um sistema de bancos de dados distribuídos, com servidores espalhados em todo o mundo. O DFNS guarda registros com endereços IP, trocas de correio e outras informações. Ao consultar um servidor DNS, um processo pode mapear um nome de domínio na Internet no endereço IP usado para comunicação com esse domínio.

O correio eletrônico é uma das duas aplicações populares para a Internet. Quase todas as pessoas, desde crianças até seus avós o utilizam agora. A maioria de sistemas de correio eletrônico no mundo emprega o sistema de correio definido nas RFCs 2821 e 2822. As mensagens enviadas nesse sistema utilizam cabeçalhos do sistema ASCII para definir propriedades de mensagens. Muitos tipos de conteúdo podem ser enviados usando o MIME. As mensagens são enviadas com o uso do SMTP, que funciona estabelecendo uma conexão TCP do host de origem até o host de destino e entregando diretamente as mensagens de correio eletrônico pela conexão TCP.

A outra aplicação importante para a Internet é a World Wide Web. A Web é um sistema para vincular documentos de hipertexto. Originalmente, cada documento

era uma página escrita em HTML, com hiperlinks para outros documentos. Hoje em dia, a XML está começando gradualmente a superar a HTML. Além disso, uma grande quantidade de conteúdo é gerada dinamicamente, com o uso de scripts do lado servidor (PHP, JSP e ASP), bem como scripts do lado cliente (em especial JavaScript). Um navegador pode exibir um documento estabelecendo uma conexão TCP para seu servidor, solicitando o documento, e depois fechando a conexão. Essas mensagens de solicitação contêm uma grande variedade de cabeçalhos para fornecer informações adicionais. Caches, replicação e redes de entrega de conteúdo são amplamente usadas para otimizar o desempenho da Web.

A Web sem fio está só começando. Os primeiros sistemas são WAP e i-mode, cada um com pequenas telas e largura de banda limitada, mas a próxima geração será mais poderosa.

A multimídia também é uma estrela que surge no universo das redes. Ela permite que os sinais de áudio e vídeo sejam digitalizados e transportados eletronicamente para exibição. O áudio exige menor largura de banda, e assim está em um estágio mais avançado. O áudio de fluxo, o rádio da Internet e a voz sobre IP são uma realidade hoje, com novas aplicações surgindo o tempo todo. O vídeo por demanda é uma área em desenvolvimento na qual existe grande interesse. Por fim, o MBone é um serviço experimental de televisão digital ao vivo de âmbito mundial transmitido pela Internet.

[T2] Problemas

1. Muitos computadores comerciais têm três identificadores distintos e exclusivos em âmbito mundial. Quais são eles?

2. De acordo com as informações fornecidas na Figura 7.3, *little-sister.cs.vu.nl* está em uma rede da classe A, B ou C?

3. Na Figura 7.3, não existe nenhum ponto depois de *rowboat*. Por que não?

4. Tente adivinhar o que o smiley :-X (às vezes, representado por :-#) poderia significar.

5. O DNS utiliza o UDP em vez do TCP. Se um pacote DNS for perdido, não haverá nenhuma recuperação automática. Isso causará um problema? Em caso afirmativo, como ele será resolvido?

6. Além de estarem sujeitos a perdas, os pacotes UDP têm um comprimento máximo, potencialmente apenas de 576 bytes. O que acontece quando um nome DNS a ser pesquisado excede esse comprimento? Ele pode ser enviado em dois pacotes?

7. Uma máquina com um único nome DNS pode ter vários endereços IP? Como isso poderia ocorrer?

8. Um computador pode ter dois nomes DNS que se enquadram em diferentes domínios de nível superior? Em caso afirmativo, forneça um exemplo plausível. Se não, explique por que não.

9. O número de empresas com um Web site cresceu de modo explosivo nos últimos anos. Como resultado, milhares de empresas se registraram no domínio *com*, provocando uma carga pesada sobre o servidor de nível superior para esse domínio. Sugira um modo de atenuar esse problema sem alterar o esquema de nomenclatura (isto é, sem introduzir novos nomes de domínios de nível superior). Sua solução pode exigir mudanças no código do cliente.

10. Alguns sistemas de correio eletrônico aceitam um campo de cabeçalho *Content Return:*. Esse campo especifica se o corpo da mensagem deve ser retornado caso não seja entregue. Esse campo pertence ao envelope ou ao cabeçalho?

11. Os sistemas de correio eletrônico precisam de diretórios para que os endereços eletrônicos das pessoas possam ser pesquisados. Para criar esses

diretórios, os nomes devem ser divididos em componentes padrão (por exemplo, primeiro nome, último nome) para possibilitar a pesquisa. Descreva alguns problemas que devem ser resolvidos para que um padrão mundial possa ser aceito.

12. O endereço de correio eletrônico de uma pessoa é seu nome de login @ nome de um domínio DNS com um registro *MX*. Os nomes de login podem ser nomes, sobrenomes, iniciais e vários outros tipos de nomes. Suponha que uma grande empresa descobrisse que estava perdendo uma grande quantidade de mensagens de correio eletrônico porque as pessoas não conheciam o nome de login do destinatário. Existe algum meio para que a empresa possa corrigir esse problema sem alterar o DNS? Nesse caso, faça uma proposta e explique como ela funciona. Caso contrário, explique por que isso é impossível.

13. Um arquivo binário tem 3072 bytes. Que tamanho ele terá se for codificado com a técnica de base64, com um par CR+LF inserido após cada 80 bytes enviados e no final?

14. Considere o esquema de codificação MIME quoted-printable. Mencione um problema não discutido no texto e proponha uma solução.

15. Cite cinco tipos MIME não listados no livro. Você poderá verificar seu navegador ou consultar a Internet para obter informações.

16. Suponha que você queira enviar um arquivo MP3 a um amigo, mas o ISP do seu amigo limita a quantidade de correio recebida a 1 MB e o arquivo MP3 tem 4 MB. Existe algum modo de lidar com essa situação usando a RFC 822 e o MIME?

17. Suponha que alguém configura um daemon de férias e depois envie uma mensagem, pouco antes de sair de férias. Infelizmente, o destinatário tirou férias por uma semana e também tem um daemon de férias instalado. O que acontece em seguida? As respostas prontas ficarão indo e voltando até alguém retornar das férias?

18. Em qualquer padrão, como o RFC 822, é necessário uma gramática exata do que é permitido, para que diferentes implementações possam interoperar. Até mesmo itens simples têm de estar definidos com cuidado. Os cabeçalhos SMTP permitem espaço em branco entre os símbolos. Forneça *duas* definições alternativas plausíveis de espaço em branco entre símbolos.

19. O daemon de férias é parte do agente do usuário ou do agente de transferência de mensagens? É claro que ele é definido com a utilização do agente do usuário, mas o agente do usuário realmente envia as respostas? Explique sua resposta.

20. O POP3 permite que os usuários busquem e baixem mensagens de correio eletrônico de uma caixa de correio remota. Isso significa que o formato interno das caixas de correio tem de ser padronizado, de forma que qualquer programa POP3 no lado cliente possa ler a caixa de correio em qualquer servidor de correio? Explique sua resposta.

21. Do ponto de vista de um ISP, o POP3 e o IMAP diferem em um aspecto importante. Em geral, os usuários do POP3 esvaziam suas caixas de correio todos os dias. Os usuários do IMAP mantêm sua correspondência no servidor indefinidamente. Imagine que você fosse chamado para aconselhar um ISP sobre que protocolo ele deveria admitir. Que considerações você faria?

22. O Webmail utiliza POP3, IMAP ou nenhum deles? Se ele utiliza algum desses protocolos, por que tal protocolo foi escolhido? Se não usa nenhum deles, qual desses protocolos tem mais afinidades com o Webmail?

23. Quando são enviadas, as páginas da Web são prefixadas por cabeçalhos MIME. Por quê?

24. Quando os visualizadores externos são necessários? Como um navegador sabe qual deles deve usar?

25. É possível um usuário clicar em um link no Netscape para abrir um auxiliar

específico e clicar no mesmo link no Internet Explorer abrindo um auxiliar completamente diferente, embora o tipo MIME retornado em ambos os casos seja idêntico? Explique sua resposta.

26. Uma servidor da Web multithreaded está organizado como mostra a Figura 7.21. Ele demora 500 μs para aceitar uma solicitação e verificar o cache. Em metade do tempo, o arquivo é encontrado no cache e retornado de imediato. Na outra metade do tempo, o módulo tem de se bloquear durante 9 ms, enquanto sua solicitação de disco é enfileirada e processada. Quantos módulos o servidor deve ter para manter a CPU ocupada o tempo todo (supondo-se que o disco não seja um gargalo)?

27. O URL padrão *http* pressupõe que o servidor da Web está escutando na porta 80. Porém, é possível para um servidor da Web escutar em alguma outra porta. Crie uma sintaxe razoável para um URL acessar um arquivo em uma porta não padronizada.

28. Embora não tenha sido mencionada no texto, uma forma alternativa para um URL é o uso do endereço IP em lugar do seu nome DNS. Um exemplo de utilização de um endereço IP é *http://192.31.231.66/index.html*. Como o navegador consegue saber se o nome que segue o esquema é um nome DNS ou um endereço IP?

29. Imagine que alguém no departamento de ciência da computação de Stanford acabou de criar um novo programa e deseja distribuí-lo por FTP. Essa pessoa coloca o programa no diretório de FTP *ftp/pub/freebies/newprog.c*. Qual deve ser o URL provável para esse programa?

30. Na Figura 7.25, *www.aportal.com* mantém as preferências do usuário em um cookie. Uma desvantagem desse esquema é que os cookies se limitam a 4 KB; assim, se as preferências forem extensas — por exemplo, muitas ações, equipes esportivas, tipos de manchetes, previsão do tempo para várias cidades, detalhes

especiais para diversas categorias de produtos e outras — o limite de 4 KB poderá ser alcançado. Crie uma forma alternativa para controlar as preferências que não tenha esse problema.

31. O Sloth Bank deseja facilitar as transações bancárias on-line para seus clientes preguiçosos, de forma que um cliente possa se associar e ser autenticado por uma senha, e o banco retorne um cookie contendo um número de identificação do cliente. Desse modo, o cliente não terá de se identificar ou digitar uma senha em visitas futuras ao banco on-line. O que você acha dessa idéia? Ela funcionará? É uma boa idéia?

32. Na Figura 7.26, o parâmetro *ALT* é definido na tag <img>. Sob que circunstâncias o navegador o utiliza, e como?

33. Em HTML, como criar uma imagem que pode ser ativada por um clique do mouse? Dê um exemplo.

34. Mostre a tag [TD]<a>[TN] necessária para tornar o string "ACM" um hiperlink para *http://www.acm.org*.

35. Crie um formulário para uma nova empresa, a Interburger. O objetivo desse formulário é permitir a encomenda de sanduíches pela Internet. O formulário deve incluir o nome do cliente, o endereço e a cidade, bem como a escolha do tamanho (gigante ou imenso) e uma opção de queijo. Os sanduíches devem ser pagos em dinheiro no momento da entrega; portanto, não são necessárias informações sobre cartões de crédito.

36. Projete um formulário para solicitar que o usuário digite dois números. Quando o usuário clicar no botão de envio, o servidor retornará a soma dos dois números. Desenvolva o lado de servidor como um script PHP.

37. Para cada uma das aplicações a seguir, informe se seria (1) possível e (2) melhor usar um script PHP ou JavaScript e por que.

(a) Exibir um calendário para qualquer mês solicitado desde setembro de 1752.

(b) Exibir a programação de vôos de Amsterdã até Nova York.

(c) Criar um grafo para representar um polinômio a partir de coeficientes fornecidos pelo usuário.

38. Escreva um programa em JavaScript que aceite um inteiro maior que 2 e diga se ele é um número primo. Observe que o JavaScript tem instruções [TD]if[TN] e [TD]while[TN] com a mesma sintaxe de C e Java. O operador de módulo é [TD]%[TN]. Se precisar da raiz quadrada de *x*, use *Math.sqrt* (*x*).

39. Aqui está uma página de HTML:

[TD]

<html> <body>

 Clique aqui para obter informações

</body> </html> [TN]

Se o usuário clicar no hiperlink, será aberta uma conexão TCP é uma série de linhas será enviada ao servidor. Liste todas as linhas enviadas.

40. O cabeçalho *If-Modified-Since* pode ser usado para verificar se uma página guardada no cache ainda é válida. Podem ser feitas solicitações de páginas contendo imagens, som, vídeo e assim por diante, bem como HTML. Você imagina que a eficiência dessa técnica é melhor ou pior para imagens JPEG, em comparação com a HTML? Pense com cuidado no que significa "eficácia" e explique sua resposta.

41. No dia de um grande evento esportivo, como o jogo do campeonato de algum esporte popular, muitas pessoas visitam o Web site oficial. Isso é um sucesso instantâneo no mesmo sentido da eleição de 2000 na Flórida? Por que ou por que não?

42. Faz sentido um único ISP funcionar como uma CDN? Nesse caso, como ele funcionaria? Se não, o que está errado com essa idéia?

43. Sob que condições o uso de uma CDN é má idéia?

44. Os terminais da Web sem fio têm baixa largura de banda, o que torna importante a codificação eficiente. Crie um esquema para transmitir texto em português de forma eficiente por um link sem fio para um dispositivo WAP. Suponha que o terminal tenha vários megabytes de ROM e uma CPU de potência moderada. *Sugestão*: Pense em como seria a transmissão em japonês, um idioma no qual cada símbolo representa uma palavra.

45. Um CD armazena 650 MB de dados. A compactação é usada em CDs convencionais (de áudio)? Explique seu raciocínio.

46. Na Figura 7.57(c), o ruído de quantização ocorre devido ao uso de amostras de 4 bits para representa 9 valores de sinais. A primeira amostra no tempo 0 é exata, mas as outras não. Qual é a porcentagem de erros para as amostras obtidas a 1/32, 2/32 e 3/32 do período?

47. Um modelo psicoacústico poderia ser usado para reduzir a largura de banda necessária para telefonia da Internet? Nesse caso, que condições, se houver, teriam de ser atendidas para fazê-lo funcionar? Se não, por que não?

48. Um servidor de fluxo de áudio está a uma distância de cerca de 50 ms em relação a um reprodutor de mídia. Ele transmite a saída a 1 Mbps. Se o reprodutor de mídia tiver um buffer de 1 MB, o que você poderá dizer sobre a posição da marca de nível baixo e da marca de nível alto?

49. O algoritmo de intercalação da Figura 7.60 tem a vantagem de poder sobreviver a um pacote perdido ocasionalmente sem introduzir um intervalo na reprodução. Porém, quando é usado para telefonia da Internet, ele também apresenta uma pequena desvantagem. Que desvantagem é essa?

50. A voz sobre IP tem os mesmos problemas com firewalls que o áudio de fluxo? Explique sua resposta.

51. Qual é a taxa de bits para transmissão de quadros em cores não compactados

de 800 × 600 pixels com 8 bits/pixel a 40 quadros por segundo?

52. Um erro de 1 bit em um quadro MPEG pode afetar outros quadros além daquele no qual o erro ocorre? Explique sua resposta.

53. Considere um servidor de vídeo com 100.000 clientes, onde cada cliente assiste a dois filmes por mês. Suponha que metade dos filmes seja transmitida às 20 horas. Quantos filmes o servidor tem de transmitir ao mesmo tempo durante esse período? Se cada filme exigir 4 Mbps, quantas conexões OC-12 o servidor precisará ter para a rede?

54. Suponha que a lei de Zipf seja válida para acessos a um servidor de vídeo com 10.000 filmes. Supondo que o servidor mantenha os 1000 filmes mais populares em disco magnético e os outros 9000 em disco óptico, forneça uma expressão que indique a fração de todas as referências feitas ao disco magnético. Crie um pequeno programa para avaliar essa expressão numericamente.

55. Alguns intrusos registraram nomes de domínios que são grafias erradas de sites corporativos comuns como, por exemplo, *www.microsfot.com*. Faça uma lista de pelo menos cinco desses domínios.

56. Várias pessoas registraram nomes DNS que consistem em *www.palavra.com*, onde *palavra* é uma palavra comum. Para cada uma das categorias a seguir, liste cinco Web sites e faça um breve resumo de sua finalidade (por exemplo, *www.estomacal.com* é um gastroenterologista de Niterói). Aqui está a lista de categorias: animais, comidas, objetos domésticos e partes do corpo. Para a última categoria, por favor tome como exemplos partes do corpo acima da cintura.

57. Projete você mesmo alguns emojis próprios usando mapas de bits de 12 × 12. Inclua namorado, namorada, professor e político.

58. Crie um servidor POP3 que aceite os comandos a seguir: *USER*, *PASS*, *LIST*, *RETR*, *DELE* e *QUIT*.

59. Reescreva o servidor da Figura 6.6 como um verdadeiro servidor da Web

usando o comando *GET* de HTTP 1.1. Ele também deve aceitar a mensagem *Host*. O servidor deve manter um cache de arquivos recentemente buscados no disco e atender a solicitações do cache, quando possível.

[TA1]Capítulo

[TA2]8

[T1]Segurança de redes

Durante as primeiras décadas de sua existência, as redes de computadores foram usadas principalmente por pesquisadores universitários, com a finalidade de enviar mensagens de correio eletrônico, e também por funcionários de empresas, para compartilhar impressoras. Sob essas condições, a segurança nunca precisou de maiores cuidados. Porém, como milhões de cidadãos comuns atualmente estão usando as redes para executar operações bancárias, fazer compras e arquivar sua devolução de impostos, a segurança das redes está despontando no horizonte como um problema potencial. Nas seções a seguir, estudaremos a segurança das redes a partir de vários ângulos, destacaremos muitas falhas e discutiremos diversos algoritmos e protocolos que tornam as redes mais seguras.

A segurança é um assunto abrangente e inclui inúmeros tipos problemas. Em sua forma mais simples, a segurança se preocupa em garantir que pessoas mal-intencionadas não leiam ou, pior ainda, modifiquem secretamente mensagens enviadas a outros destinatários. Outra preocupação da segurança são as pessoas que tentam ter acesso a serviços remotos que elas não estão autorizadas a usar. Ela também lida com meios para saber se uma mensagem supostamente verdadeira é um trote. A segurança trata de situações em que mensagens legítimas são capturadas e reproduzidas, além de lidar com pessoas que tentam negar o fato de terem enviado determinadas mensagens.

A maior parte dos problemas de segurança é causada intencionalmente por pessoas maliciosas que tentam obter algum benefício, chamar a atenção ou prejudicar alguém. Alguns dos invasores mais comuns estão listados na Figura 8.1. A partir dessa lista fica claro que tornar uma rede segura envolve muito mais

do que simplesmente mantê-la livre de erros de programação. Para tornar uma rede segura, com freqüência é necessário lidar com adversários inteligentes, dedicados e, às vezes, muito bem subsidiados. Você também deverá ter em mente que as medidas utilizadas para interromper a atividade de adversários eventuais terão pouco impacto sobre os adversários "mais espertos". Os registros policiais mostram que a maioria dos ataques não é perpetrada por estranhos que grampeiam uma linha telefônica, mas por pessoas ressentidas com a organização a que pertencem. Conseqüentemente, os sistemas de segurança devem ser projetados tendo em vista esse fato.

[arte: ver original p. 722]

[T]Tabela

Adversário	**Objetivo**
Estudante	Divertir-se bisbilhotando as mensagens de correio eletrônico de outras pessoas
Cracker	Testar o sistema de segurança de alguém; roubar dados
Representante de vendas	Tentar representar toda a Europa e não apenas Andorra
Executivo	Descobrir a estratégia de marketing do concorrente
Ex-funcionário	Vingar-se por ter sido demitido
Contador	Desviar dinheiro de uma empresa
Corretor de valores	Negar uma promessa feita a um cliente através de uma mensagem de correio eletrônico
Vigarista	Roubar números de cartão de crédito e vendê-los
Espião	Descobrir segredos militares ou industriais de um inimigo
Terrorista	Roubar segredos de armas bacteriológicas

[F]Figura 8.1

[FL] Algumas pessoas que podem causar problemas de segurança e os motivos para fazê-lo

Os problemas de segurança das redes podem ser divididos nas seguintes áreas interligadas: sigilo, autenticação, não repúdio e controle de integridade. O sigilo está relacionado ao fato de manter as informações longe de usuários não autorizados. É isso que geralmente nos vem à mente quando pensamos em segurança de redes. Em geral, a autenticação cuida do processo de determinar com quem você está se comunicando antes de revelar informações sigilosas ou entrar em uma transação comercial. O não repúdio trata de assinaturas: como provar que seu cliente realmente fez um pedido eletrônico de 10 milhões de unidades de um produto com preço unitário de 89 centavos quando mais tarde ele afirmar que o preço era 69 centavos? Ou então, é possível que ele afirme que nunca efetuou qualquer pedido. Por fim, como você pode se certificar de que uma mensagem recebida é realmente legítima e não algo que um oponente mal-intencionado modificou ou inventou?

Todas essas questões (sigilo, autenticação, não repúdio e controle de integridade) também ocorrem em sistemas tradicionais, mas com algumas diferenças significativas. O sigilo e a integridade são obtidos através da utilização de correspondência registrada e do bloqueio de documentos. Agora é mais difícil roubar o trem postal que nos tempos de Jesse James.

Além disso, normalmente as pessoas conseguem distinguir um documento original de uma fotocópia, e isso freqüentemente faz diferença para elas. Como teste, tire uma fotocópia de um cheque válido. Tente descontar o cheque original na segunda-feira. Agora tente descontar a fotocópia do cheque na terça-feira. Observe a diferença no comportamento do caixa. Com os cheques eletrônicos, você não tem como distinguir entre o original e a cópia. Talvez leve algum tempo até os bancos se acostumarem com isso.

As pessoas autenticam outras pessoas ao reconhecer seus rostos, vozes e

caligrafia. As comprovações de assinatura são feitas através de assinaturas em papel timbrado, de símbolos em alto relevo etc. Em geral, as falsificações podem ser detectadas por especialistas em caligrafia, papel e tinta. Nenhuma dessas opções está disponível eletronicamente. É claro que são necessárias outras soluções.

Antes de entrarmos nas soluções propriamente ditas, vale a pena dedicar alguns momentos considerando a que parte da pilha de protocolos pertence a segurança de redes. Provavelmente não existe uma parte específica. Todas as camadas contribuem de alguma forma. Na camada física, os "grampos" podem ser anulados mantendo-se as linhas de transmissão em tubos lacrados contendo gás em alta pressão. Qualquer tentativa de perfurar um tubo liberará o gás, reduzindo a pressão e disparando um alarme. Alguns sistemas militares utilizam essa técnica.

Na camada de enlace de dados, os pacotes de uma linha ponto a ponto podem ser codificados à medida que saem de uma máquina, e decodificados quando entram em outro sistema. Todos os detalhes podem ser tratados na camada de enlace de dados, com as camadas mais altas alheias ao que está acontecendo. No entanto, essa solução se mostra ineficiente quando os pacotes têm de atravessar vários roteadores, pois é necessário descriptografar os pacotes em cada roteador, o que os torna vulneráveis a ataques dentro do roteador. Além disso, essa estratégia não permite que algumas sessões sejam protegidas (como, por exemplo, aquelas que envolvem compras on-line por cartão de crédito) e outras não. Todavia, a **criptografia de enlace**, como esse método é chamado, pode ser facilmente incluída em qualquer rede e com freqüência é muito útil.

Na camada de rede, podem ser instalados firewalls para manter ou descartar pacotes. A segurança do IP também funciona nessa camada.

Na camada de transporte, é possível criptografar conexões inteiras fim a fim, ou

seja, processo a processo. Para obter segurança máxima, é necessário que ela seja fim a fim.

Finalmente, questões como autenticação do usuário e não repúdio só podem ser tratadas na camada de aplicação.

Tendo em vista que a segurança não se ajusta nitidamente a qualquer camada, ela também não se enquadra em nenhum capítulo deste livro. Por essa razão, vamos estudá-la separadamente em seu próprio capítulo.

Embora este capítulo seja longo, técnico e essencial ele também é quase irrelevante no momento. Por exemplo, está bem documentado o fato de que a maioria das falhas de segurança em bancos se deve a funcionários incompetentes, procedimentos de segurança negligentes ou fraudes internas, e não a criminosos inteligentes que grampeiam linhas telefônicas e depois decodificam mensagens criptografadas. Se uma pessoa puder entrar em uma agência bancária qualquer com um tira de extrato de caixa eletrônico que encontrou na rua, afirmando que esqueceu número de identificação e receber um novo PIN (personal identification number — número de identificação pessoal) na mesma hora (em nome das boas relações com os clientes), nem toda criptografia do mundo evitará o abuso. Sobre esse aspecto, o livro de Ross Anderson é um excelente alerta, pois documenta centenas de exemplos de falhas de segurança em numerosas indústrias, quase todas causadas por aquilo que se poderia chamar educadamente de práticas comerciais relaxadas ou desatenção com a segurança (Anderson, 2001). Apesar disso, somos otimistas e acreditamos que, à medida que o comércio eletrônico se tornar mais difundido, as empresas irão depurar seus procedimentos operacionais, eliminando esse furo e trazendo os aspectos técnicos da segurança de volta à cena.

Com exceção da segurança na camada física, quase toda segurança se baseia em princípios criptográficos. Por essa razão, começaremos nosso estudo da

segurança examinando em detalhes a criptografia. Na Seção 8.1, veremos alguns princípios básicos. Da Seção 8.2 até a Seção 8.5, examinaremos alguns algoritmos e estruturas de dados fundamentais usados em criptografia. Em seguida, examinaremos em detalhes como esses conceitos podem ser usados para se alcançar a segurança em redes. Concluiremos com alguns conceitos breves sobre tecnologia e sociedade.

Antes de começarmos, devemos chamar a atenção para o que não é abordado neste capítulo. Procuramos nos concentrar em questões de redes, e não em questões relacionadas ao sistema operacional e às aplicações, embora seja difícil traçar a linha que separa esses assuntos. Por exemplo, não há nada aqui sobre autenticação do usuário com a utilização da biometria, segurança de senhas, ataques de estouro de buffers, cavalos de Tróia, spoofing de login, bombas lógicas, vírus, vermes e temas semelhantes. Todos esses tópicos são abordados detalhadamente no Capítulo 9 do livro *Modern Operating Systems* (Tanenbaum, 2001). O leitor interessado deve consultar esse livro para conhecer os aspectos de segurança relacionados aos sistemas. Agora, vamos iniciar nossa jornada.

[T2] 8.1 Criptografia

A palavra **criptografia** vem das palavras gregas que significam "escrita secreta". A criptografia tem uma longa e interessante história de milhares de anos. Nesta seção, vamos esquematizar alguns destaques, que serão usados como informações básicas para o que vem a seguir. Se desejar um histórico completo da criptografia, recomendamos a leitura do livro de Khan (1995). Para ver um tratamento completo do estado da arte atual em segurança e algoritmos criptográficos, protocolos e aplicações, consulte (Kaufman *et al.*, 2002). Para uma abordagem mais matemática, consulte (Stinson, 2002). Se preferir uma abordagem menos matemática, consulte (Burnett e Paine, 2001).

Os profissionais fazem distinção entre cifras e códigos. Uma **cifra** é uma transformação de caractere por caractere ou de bit por bit, sem levar em conta a estrutura lingüística da mensagem. Em contraste, um código substitui uma palavra por outra palavra ou símbolo. Os códigos não são mais utilizados, embora tenham uma história gloriosa. O código mais bem-sucedido já inventado foi usado pelas forças armadas dos Estados Unidos durante a Segunda Guerra Mundial no Pacífico. Eles simplesmente tinham índios Navajo que se comunicam uns com os outros usando palavras Navajo específicas para termos militares como, por exemplo, *chay-dagahi-nail-tsaidi* (literalmente assassino de cágado) para indicar uma arma antitanque. A linguagem Navajo é altamente tonal, extremamente complexa, e não tem nenhuma forma escrita. Além disso, nem uma única pessoa no Japão conhecia nada sobre ela.

Em setembro de 1945, o *San Diego Union* descreveu o código da seguinte forma: "Por três anos, onde quer que os Marines aterrissassem, os japoneses recebiam uma enxurrada de estranhos ruídos gorgolejantes entremeados com outros sons que lembravam o clamor de um monge tibetano e o som de uma bolsa de água quente sendo esvaziada". Os japoneses nunca conseguiram romper o código e muitos índios Navajo receberam altas honras militares por serviço e bravura extraordinários. O fato dos Estados Unidos terem conseguido romper o código japonês e os japoneses nunca terem conseguido quebrar o código Navarro desempenhou um papel crucial nas vitórias americanas no Pacífico.

[T3] 8.1.1 Introdução à criptografia

Historicamente, quatro grupos de pessoas utilizaram e contribuíram para a arte da criptografia: os militares, os diplomatas, as pessoas que gostam de guardar memórias e os amantes. Dentre eles, os militares tiveram o papel mais importante e definiram as bases para a tecnologia. Dentro das organizações

militares, tradicionalmente as mensagens a serem criptografadas são entregues a auxiliares mal remunerados que se encarregam de criptografá-las e transmiti-las. O grande volume de mensagens impedia que esse trabalho fosse feito por alguns poucos especialistas.

Até o advento dos computadores, uma das principais restrições impostas à criptografia era a habilidade do auxiliar de criptografia fazer as transformações necessárias, em geral com poucos equipamentos e no campo de batalha. Uma outra restrição era a dificuldade de alternar os métodos criptográficos rapidamente, pois isso exigia a repetição do treinamento de um grande número de pessoas. No entanto, o perigo de um auxiliar de criptografia ser capturado pelo inimigo tornou indispensável a possibilidade de se alterar o método criptográfico instantaneamente, se necessário. Essas necessidades conflitantes fizeram surgir o modelo da Figura 8.2.

[arte: ver original p. 725]

[Dísticos]

[1] Intruso passivo apenas escuta

[2] Intruso

[3] Intruso ativo pode alterar mensagens

[4] Texto simples, P Método de criptografia, E Método de descriptografia, D Texto simples, P

[5] Texto cifrado, C = $E_K(P)$

[6] Chave de criptografia, K

[7] Chave de descriptografia, K

[F]Figura 8.2

[FL] O modelo de criptografia (para uma cifra de chave simétrica)

As mensagens a serem criptografadas, conhecidas como **texto simples**, são

transformadas por uma função que é parametrizada por uma **chave.** Em seguida, a saída do processo de criptografia, conhecida como **texto cifrado**, é transmitida, normalmente através de um mensageiro ou por rádio. Presumimos que o inimigo, ou **intruso**, ouça e copia cuidadosamente o texto cifrado completo. No entanto, ao contrário do destinatário pretendido, ele não conhece a chave para descriptografar o texto e, portanto, não pode fazê-lo com muita facilidade. Às vezes, o intruso pode não só escutar o que se passa no canal de comunicação (intruso passivo), como também pode gravar mensagens e reproduzi-las mais tarde, injetar suas próprias mensagens ou modificar mensagens legítimas antes que elas cheguem ao receptor (intruso ativo). A arte de solucionar mensagens cifradas é chamada **criptoanálise.** A arte de criar mensagens cifradas (criptografia) e solucioná-las (criptoanálise) é chamada coletivamente **criptologia.**

Com freqüência, será útil e prático ter uma notação para estabelecer uma relação entre o texto simples, o texto cifrado e as chaves. Utilizaremos $C = E_K(P)$ para denotar que a criptografia do texto simples P usando a chave K gera o texto cifrado C. Da mesma forma, $P = D_K(C)$ representa a descriptografia de C para se obter o texto simples outra vez. Então, temos:

$D_K(E_K(P)) = P$

Essa notação sugere que E e D são simplesmente funções matemáticas, o que é verdade. A única parte complicada é que ambas são funções de dois parâmetros, e escrevemos um desses parâmetros (a chave) como um caractere subscrito, em vez de representá-lo como um argumento, para distingui-lo da mensagem.

Uma regra fundamental da criptografia é que se deve supor que o criptoanalista conhece os métodos genéricos de criptografia e descriptografia que são utilizados. Em outras palavras, o criptoanalista sabe como funciona o método de criptografia, E, e o método de descriptografia D da Figura 8.2. O esforço necessário para criar, testar e instalar um novo algoritmo toda vez que o antigo

método (supostamente) é comprometido sempre dificultou a manutenção desse segredo. Imaginar que o algoritmo de criptografia é secreto quando ele não é resulta em mais prejuízo do que em benefícios.

É nesse ponto que entra a chave. A chave consiste em um string (relativamente) curto que seleciona uma das muitas formas possíveis de criptografia. Ao contrário do método genérico, que só pode ser modificado a cada período de alguns anos, a chave pode ser alterada sempre que necessário. Portanto, nosso modelo básico é um método genérico publicamente conhecido, parametrizado por uma chave secreta que pode ser alterada com facilidade. A idéia de que o criptoanalista conhece os algoritmos e que o segredo reside exclusivamente nas chaves é chamada **princípio Kerckhoff**, que recebeu esse nome em homenagem ao criptógrafo militar flamengo Auguste Kerckhoff que e enunciou primeiro em 1883 (Kerckhoff, 1883). Desse modo, temos:

Princípio de Kerckhoff: Todos os algoritmos devem ser públicos; apenas as chaves são secretas

Devemos enfatizar o caráter não sigiloso do algoritmo. Tentar manter o algoritmo secreto, uma estratégia conhecida no ramo como **segurança pela obscuridade**, nunca funciona. Além disso, ao tornar o algoritmo público, o especialista em criptografia se livra de te de consultar inúmeros criptólogos ansiosos por decodificar o sistema para poderem publicar artigos demonstrando sua esperteza e inteligência. Caso muitos especialistas tenham tentado decodificar o algoritmo durante cinco anos após sua publicação e nenhum tenha obtido sucesso, isso provavelmente significa que o algoritmo é sólido.

Na verdade, o sigilo está na chave, e seu tamanho é uma questão muito importante do projeto. Considere um bloqueio de combinação simples. Segundo o princípio geral, você insere dígitos em seqüência. Todo mundo sabe disso, mas a chave é secreta. Uma chave com um tamanho de dois dígitos significa que

existem 100 possibilidades, uma chave de três dígitos significa mil possibilidades e uma chave de seis dígitos significa um milhão de possibilidades. Quanto maior for a chave, mais alto será o **fator de trabalho** com que o criptoanalista terá de lidar. O fator de trabalho para decodificar o sistema através de uma exaustiva pesquisa no espaço da chave é exponencial em relação ao tamanho da chave. O sigilo é decorrente da presença de um algoritmo forte (mas público) e de uma chave longa. Para impedir que o seu irmãozinho leia suas mensagens de correio eletrônico, serão necessárias chaves de 64 bits. Para uso comercial de rotina, devem ser usados pelo menos 128 bits. Para manter o governo de outros países à distância, são necessárias chaves de pelo menos 256 bits, de preferência maiores.

Do ponto de vista do criptoanalista, o problema da criptoanálise apresenta três variações principais. Quando tem um determinado volume de texto cifrado mas nenhum texto simples, o analista é confrontado com o problema de haver **somente texto cifrado**. Os criptogramas da seção de palavras cruzadas do jornal são um exemplo desse tipo de problema. Quando há uma correspondência entre o texto cifrado e o texto simples, o problema passa a ser chamado **texto simples conhecido**. Por fim, quando o criptoanalista tem a possibilidade de codificar trechos do texto simples escolhidos por ele mesmo, temos o problema do **texto simples escolhido**. Os criptogramas dos jornais poderiam ser decodificados de forma trivial se o criptoanalista tivesse a permissão de fazer perguntas tais como: Qual é a criptografia de ABCDEFGHIJKL?

Com freqüência, os novatos na área de criptografia pressupõem que, se uma cifra puder resistir a uma estratégia de texto cifrado, isso significa que ela é segura. Essa suposição é muito ingênua. Em muitos casos, o criptoanalista pode fazer uma estimativa com base em trechos do texto simples. Por exemplo, a primeira mensagem que muitos sistemas de tempo compartilhado emitem quando você os

chama é "LOGIN:". Equipado com alguns pares de texto simples/texto cifrado, o trabalho do criptoanalista se torna muito mais fácil. Para obter segurança, o autor da criptografia deve ser conservador e se certificar de que o sistema seja inviolável, mesmo que seu oponente seja capaz de criptografar o texto simples escolhido.

Historicamente, os métodos de criptografia têm sido divididos em duas categorias: as cifras de substituição e as cifras de transposição. Em seguida, trataremos de cada uma dessas técnicas como informações básicas para a criptografia moderna.

[T3] 8.1.2 Cifras de substituição

Em uma **cifra de substituição**, cada letra ou grupo de letras é substituído por outra letra ou grupo de letras, de modo a criar um "disfarce". Uma das cifras mais antigas é a **cifra de César**, atribuída a Júlio César. Nesse método, *a* se torna *D*, *b* se torna *E*, *c* se torna *F*, ... e *z* se torna *C*. Por exemplo, *ataque* passaria a ser *DWDTXH*. Nos exemplos, o texto simples é apresentado em letras minúsculas e o texto cifrado em letras maiúsculas.

Uma ligeira generalização da cifra de César permite que o alfabeto do texto cifrado seja deslocado *k* letras, em vez de 3. Nesse caso, *k* passa a ser uma chave para o método genérico dos alfabetos deslocados em forma circular. A cifra de César pode ter enganado os cartagineses, mas nunca mais enganou ninguém.

O próximo aprimoramento é fazer com que cada um dos símbolos do texto simples, digamos 26 letras, seja mapeado para alguma outra letra. Por exemplo,

texto simples: a b c d e f g h i j k l m n o p q r s t u v w x y z

texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Esse sistema geral é chamado **substituição monoalfabética**, sendo a chave o string de 26 letras correspondente ao alfabeto completo. Para a chave anterior, o

texto simples *ataque* seria transformado no texto cifrado *QZQJXT*.

À primeira vista, talvez esse sistema pareça seguro, pois apesar de conhecer o sistema genérico (substituição de letra por letra), o criptoanalista não sabe qual das $26! \approx 4 \times 10^{26}$ chaves possíveis está em uso. Ao contrário do que acontece com a cifra de César, experimentar todas elas não é uma estratégia muito interessante. Mesmo a 1 ns por solução, um computador levaria 10^{10} anos para experimentar todas as chaves.

Todavia, com um volume de texto cifrado surpreendentemente pequeno, a cifra pode ser descoberta com facilidade. A estratégia básica se beneficia das propriedades estatísticas dos idiomas. Por exemplo, em inglês *e* é a letra mais comum, seguida de *t*, *o*, *a*, *n*, *i* etc. As combinações de duas letras, ou **digramas**, mais comuns são *th*, *in*, *er*, *re* e *an*. As combinações de três letras, ou **trigramas**, mais comuns são *the*, *ing*, *and* e *ion*.

Um criptoanalista que esteja tentando decodificar uma cifra monoalfabética começaria contando as freqüências relativas de todas as letras do texto cifrado. Depois disso, através de tentativas, ele atribuiria *e* à letra mais comum e *t* à próxima letra mais comum. Em seguida, verificaria os trigramas para encontrar um no formato *tXe*, o que poderia sugerir que *X* é *h*. Da mesma forma, se o padrão *thYt* ocorrer com freqüência, provavelmente isso significará que *Y* representa *a*. Com essas informações, o criptoanalista poderá procurar por um trigrama com o formato *aZW* que ocorra com freqüência (muito provavelmente *and*). Fazendo estimativas em relação a digramas, trigramas e letras mais comuns, e conhecendo os prováveis padrões de vogais e consoantes, o criptoanalista criaria um texto simples através de tentativas, letra por letra.

Outra estratégia é adivinhar uma palavra ou frase provável. Por exemplo, considere o seguinte texto cifrado de uma empresa de contabilidade (montado em grupos de cinco caracteres):

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ

QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ

DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

Nos Estados Unidos, uma palavra muito provável em uma mensagem de uma empresa de contabilidade é *financial*. Utilizando nosso conhecimento de que *financial* tem um caractere repetido (*i*), com quatro outras letras entre suas ocorrências, estamos procurando letras repetidas no texto cifrado com esse espaço entre elas. Encontramos 12 casos como esse nas posições 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 e 82. No entanto, apenas dois deles, 31 e 42, têm a letra seguinte (que corresponde a *n* no texto simples) repetida na localização adequada. Dessas duas, apenas 31 também tem a letra *a* corretamente posicionada; portanto, sabemos que *financial* começa na posição 30. Desse ponto em diante, fica fácil deduzir a chave utilizando a estatística de freqüência para o texto em inglês.

[T3] 8.1.3 Cifras de transposição

As cifras de substituição preservam a ordem dos símbolos no texto simples, mas disfarçam esses símbolos. Por outro lado, as **cifras de transposição** reordenam as letras, mas não as disfarçam. A Figura 8.3 mostra uma cifra de transposição muito comum, a transposição de colunas. A cifra se baseia em uma chave que é uma palavra ou frase que não contém letras repetidas. Nesse exemplo, MEGABUCK é a chave. O objetivo da chave é numerar as colunas de modo que a coluna 1 fique abaixo da letra da chave mais próxima do início do alfabeto e assim por diante. O texto simples é escrito horizontalmente, em linhas. O texto cifrado é lido em colunas, a partir da coluna cuja letra da chave seja a mais baixa.

[arte: ver original p. 729]

[Dísticos]

[1]

M E G A B U C K

7 4 5 1 2 8 3 6

p l e a s e t r Texto simples

a n s f e r o n pleasetransferonemilliondollarsto

e m i l l i o n myswissbankaccountsixtwotwo

d o l l a r s t Texto cifrado

o m y s w i s s AFLLSKSOSELAWAIATOOSSCTCLNMOMANT

b a n k a c c o ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

u n t s i x t w

o t w o a b c d

[F]Figura 8.3

[FL] Uma cifra de transposição

Para romper uma cifra de transposição, o criptoanalista deve primeiro estar ciente de que está lidando com uma cifra de transposição. Examinando a freqüência de *E*, *T*, *A*, *O*, *I*, *N* etc., fica fácil constatar se essas letras se encaixam no padrão normal para texto simples. Se houver correspondência, isso significa que a cifra é evidentemente uma cifra de transposição, pois nesse tipo de cifra cada letra é representada por ela mesma, mantendo intacta a distribuição de freqüências.
A próxima etapa é fazer uma estimativa do número de colunas. Em muitos casos, uma palavra ou frase provável pode ser deduzida a partir do contexto da mensagem. Por exemplo, suponha que o nosso criptoanalista tenha suspeitado de que a frase em texto simples *milliondollars* ocorre em algum lugar na mensagem. Observe que os digramas *MO*, *IL*, *LL*, *LA*, *IR* e *OS* ocorrem no texto cifrado como um resultado do desdobramento dessa frase. No texto cifrado, a letra *O* vem depois da letra *M* (ou seja, elas são verticalmente adjacentes na

coluna 4), pois estão separadas na provável frase por uma distância igual ao tamanho da chave. Se tivesse sido usada uma chave de tamanho sete, teriam surgido os digramas *MD*, *IO*, *LL*, *LL*, *IA*, *OR* e *NS*. Na verdade, para cada tamanho de chave, é produzido um conjunto de digramas diferente no texto cifrado. Ao tentar encontrar diferentes possibilidades, muitas vezes o criptoanalista é capaz de determinar com facilidade o tamanho da chave.

A última etapa é ordenar as colunas. Quando o número de colunas k é pequeno, cada um dos $k(k - 1)$ pares de colunas pode ser examinado para que seja constatado se suas freqüências de digramas correspondem às do texto simples em inglês. O par que tiver a melhor correspondência será considerado na posição correta. Em seguida, cada uma das colunas restantes é experimentada como sucessora desse par. A coluna cujas freqüências de digramas e trigramas proporcione a melhor correspondência será experimentalmente considerada correta. O processo inteiro continua até ser encontrada uma ordenação potencial. O mais provável é que o texto simples seja reconhecido nesse ponto (por exemplo, se ocorrer *milloin*, ficará claro qual é o erro).

Algumas cifras de transposição aceitam um bloco de tamanho fixo como entrada e produzem um bloco de tamanho fixo como saída. Essas cifras podem ser completamente descritas fornecendo-se uma lista que informe a ordem na qual os caracteres devem sair. Por exemplo, a cifra da Figura 8.3 pode ser vista como uma cifra de blocos de 64 caracteres. Sua saída é 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, ..., 62. Em outras palavras, o quarto caractere de entrada, *a*, é o primeiro a sair, seguido pelo décimo segundo, *f*, e assim por diante.

[T3] 8.1.4 Chave única

Na verdade, é fácil criar uma cifra inviolável; a técnica é conhecida há décadas. Primeiro, escolha como chave um string de bits aleatórios. Em seguida, converta

o texto simples em um string de bits, utilizando por exemplo sua representação ASCII. Por fim, calcule o OR exclusivo (XOR) desses dois strings. O texto cifrado resultante não pode ser violado porque, em uma amostra suficientemente grande de texto cifrado, cada letra ocorrerá com a mesma freqüência, bem como digrama, cada trigrama e assim por diante. Esse método, conhecido como **chave única**, é imune a todos os ataques presentes e futuros, quanta capacidade computacional tenha o intruso. A razão deriva da teoria da informação: simplesmente não existe nenhuma informação na mensagem, todos os textos simples possíveis com o tamanho dado são igualmente prováveis.

Um exemplo de como as chaves únicas são usadas, é dado na Figura 8.4. Primeiro, a mensagem 1, "I love you", é convertida em ASCII de 7 bits. Em seguida, uma chave única chamada chave 1, é escolhida e sujeita à operação XOR com a mensagem para se obter o texto cifrado. Um criptoanalista poderia experimentar todas as chaves únicas possíveis para ver que texto resultou para cada uma. Por exemplo, a chave única listada como chave 2 na figura poderia ser experimentada, resultando no texto simples 2, "Elvis lives", que pode ser ou não plausível (um assunto que está além do escopo deste livro). De fato, para cada texto simples ASCII de 11 caracteres, existe uma chave única que o gera. É isso que queremos dizer quando mencionamos que não existe nenhuma informação no texto cifrado: é possível obter qualquer mensagem com o tamanho correto a partir dele.

[arte: ver original p. 731]

[T]Tabela

Mensagem 1: 1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110

Chave 1: 1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011

Texto cifrado: 0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Chave 2: 1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110

Texto simples 2: 1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

[F]Figura 8.4

[FL] O uso de uma chave única para criptografia e a possibilidade de conseguir qualquer texto simples que seja possível a partir do texto cifrado pela utilização de alguma outra chave

As chaves únicas são ótimas na teoria, mas têm várias desvantagens na prática. Para começar, a chave não pode ser memorizada; então, tanto o remetente quanto o destinatário devem levar uma cópia escrita com eles. Se qualquer um dos dois estiver sujeito à possibilidade de captura, as chaves escritas sem dúvida serão indesejáveis. Além disso, a quantidade total de dados que podem ser transmitidos é limitada pelo tamanho da chave disponível. Se o espião tiver muita sorte e descobrir uma grande quantidade de dados, talvez ele seja incapaz de transmiti-los de volta para a matriz, porque a chave foi consumida. Outro problema é a sensibilidade do método para caracteres perdidos ou inseridos. Se o transmissor e o receptor ficarem fora de sincronismo, todos os dados a partir desse momento parecerão adulterados.

Com o advento dos computadores, a chave única se tornou potencialmente prática para algumas aplicações. A origem da chave poderia ser um DVD especial contendo vários gigabytes de informações que, se transportadas em uma caixa de filmes em DVD e tivessem no início alguns minutos de vídeo, nem sequer seriam suspeitas. É claro que, nas redes de gigabits, ter de inserir um novo DVD a cada

30 segundos seria algo tremendamente entediante. Além disso, os DVDs devem ser transportados pessoalmente do transmissor para o receptor, antes de ser possível enviar qualquer mensagem, o que reduz bastante sua utilidade prática.

[T4] Criptografia quântica

É interessante observar que talvez haja uma solução para o problema de como transmitir a chave única pela rede, e ela vem de uma fonte muito improvável: a mecânica quântica. Essa área ainda é experimental, mas os testes iniciais são promissores. Se eles puderem ser aperfeiçoados e se tornarem eficientes, quase toda a criptografia será realizada eventualmente com a utilização de chaves únicas, pois elas talvez sejam seguras. A seguir, explicaremos em linhas gerais como funciona esse método, denominado **criptografia quântica.** Em particular, descreveremos um protocolo chamado **BB84** para indicar seus autores e o ano da publicação (Bennet e Brassard, 1984).

Uma usuária chamada Alice quer estabelecer uma chave única com um segundo usuário, Bob. Alice e Bob são chamados **protagonistas**, os personagens principais de nossa história. Por exemplo, Bob é um banqueiro com quem Alice gostaria de realizar negócios. Os nomes "Alice" e "Bob" foram usados como protagonistas em praticamente todos os ensaios e livros sobre criptografia na última década. Os criptógrafos amam a tradição. Se fôssemos usar "Andy" e "Barbara" como protagonistas, ninguém acreditaria em nada do que fosse explicado neste capítulo. Então, que seja!

Se Alice e Bob pudessem estabelecer uma chave única, eles teriam a possibilidade de empregá-la para se comunicarem com segurança. A pergunta é: como eles podem estabelecê-la sem anteriormente troca DVDs? Podemos supor que Alice e Bob estão em extremidades opostas de um cabo de fibra óptica pelo qual podem enviar e receber pulsos de luz. Porém, uma intrépida intrusa chamada Trudy pode

cortar a fibra e criar um grampo ativo. Trudy pode ler todos os bits em ambos os sentidos. Ela também pode enviar falsas mensagens nos dois sentidos. A situação pode parecer desesperada para Alice e Bob, mas a criptografia quântica pode trazer uma nova luz sobre o assunto.

A criptografia quântica se baseia no fato de que a luz se propaga em pequenos pacotes chamados **fótons**, que apresentam algumas propriedades peculiares. Além disso, a luz pode ser polarizada ao passar por um filtro de polarização, um fato bem conhecido para os usuários de óculos de sol e fotógrafos. Se um feixe de luz (isto é, um fluxo de fótons) passar por um filtro de polarização, todos os fótons que emergirem dele serão polarizados na direção do eixo do filtro (por exemplo, vertical). Se o feixe passar agora por um segundo filtro de polarização, a intensidade da luz que emergirá do segundo filtro será proporcional ao quadrado do cosseno do ângulo entre os eixos. Se os dois eixos forem perpendiculares, nenhum fóton passará pelo filtro. A orientação absoluta dos dois filtros não importa; só interessa o ângulo entre seus eixos.

Para gerar uma chave única, Alice precisa de dois conjuntos de filtros de polarização. O primeiro conjunto consiste em um filtro vertical e um filtro horizontal. Essa escolha é chamada **base retilínea.** Uma base é apenas um sistema de coordenadas. O segundo conjunto de filtros é idêntico, exceto por estar deslocado 45 graus, de forma que um filtro abrange desde o canto inferior esquerdo até o canto superior direito, e o outro filtro abrange desde o canto superior esquerdo até o canto inferior direito. Essa escolha é chamada **base diagonal.** Desse modo, Alice tem duas bases, que ela pode inserir rapidamente em seu feixe à vontade. Na realidade, Alice não tem quatro filtros separados, mas um cristal, cuja polarização pode ser trocada eletricamente para qualquer das quatro direções permitidas, em alta velocidade. Bob tem o mesmo equipamento de Alice. O fato de Alice e Bob terem cada um duas bases disponíveis é essencial

para a criptografia quântica.

Para cada base, Alice atribui agora uma direção como 0 e a outra como 1. No exemplo apresentado a seguir, supomos que ela escolhe a direção vertical como 0 e a horizontal como 1. Independentemente, ela também escolhe do canto inferior esquerdo até o canto superior direito como 0, e do canto superior esquerdo até o canto inferior direito como 1. Alice envia essas escolhas a Bob como texto simples.

Agora, Alice escolhe uma chave única, por exemplo com base em um gerador de números aleatórios (um assunto por si só bastante complexo). Ela o transfere bit por bit para Bob, escolhendo uma de suas bases ao acaso para cada bit. Para enviar um bit, sua pistola de fótons emite um fóton polarizado de maneira apropriada, conforme a base que ela está usando para esse bit. Por exemplo, ela poderia escolher as bases diagonal, retilínea, retilínea, diagonal, retilínea etc. Para enviar sua chave única igual a 1001110010100110 com essas bases, ela enviaria os fótons mostrados na Figura 8.5(a). Dada a chave única e a seqüência de bases, a polarização a ser usada para cada bit é determinada de forma exclusiva. Bits enviados um fóton de cada vez são chamados **qubits**.

[arte: ver original p. 733]

[Dísticos]

[1] Número do bit 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

[2] Dados 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0

[3] (a) O que Alice envia

(b) Bases de Bob

(c) O que Bob recebe

[4] (d) Não Sim Não Sim Não Não Não Sim Sim Não Sim Sim Sim Não Sim Não Base correta?

[5] (e) 0 1 0 1 1 0 0 1 Chave única

[6] (f) Bases de Trudy

[7] (g) x 0 x 1 x x x ? 1 x ? ? 0 x ? x Chave de Trudy

[F]Figura 8.5

[FL] Um exemplo de criptografia quântica

Bob não sabe que base usar, e assim escolhe uma base ao acaso para cada fóton que chega e simplesmente o utiliza, como mostra a Figura 8.5(b). Se escolher a base correta, ele receberá o bit correto. Se escolher a base incorreta, ele receberá um bit aleatório porque, se um fóton acessar um filtro polarizado a 45 graus em relação à sua própria polarização, ele saltará ao acaso para a polarização do filtro ou para uma polarização perpendicular à do filtro, com igual probabilidade. Essa propriedade dos fótons é fundamental para a mecânica quântica. Desse modo, alguns bits estão corretos e alguns são aleatórios, mas Bob não consegue distingui-los. Os resultados de Bob estão representados na Figura 8.5(c).

De que maneira Bob pode descobrir quais são as bases corretas e quais são as erradas entre as que recebeu? Ele simplesmente diz a Alice que base usou para cada bit em texto simples, e ela lhe diz quais são as bases corretas e quais são as erradas em texto simples, como mostra a Figura 8.5(d). A partir dessas informações, ambos podem construir um string de bits com os palpites corretos, como mostra a Figura 8.5(e). Em média, esse string de bits terá metade do comprimento do string de bits original mas, como ambas as partes o conhecem, elas poderão usá-lo como uma chave única. Tudo que Alice tem a fazer é transmitir um string de bits um pouco maior que o dobro do tamanho desejado, para que ela e Bob tenham uma chave única com o comprimento apropriado. Problema resolvido.

Porém, espere um minuto. Esquecemos de Trudy. Vamos supor que ela esteja curiosa para saber o que Alice tem a dizer e corte o cabo de fibra, inserindo seu

próprio detector e transmissor. Infelizmente para Trudy, ela também não sabe que base usar para cada fóton. O melhor que ela pode fazer é escolher uma base ao acaso para cada um dos fótons, como fez Bob. Um exemplo de suas escolhas é mostrado na Figura 8.5(f). Quando mais tarde Bob informar (em texto simples) que bases usou e Alice disser a ele (em texto simples) quais delas estão corretas, Trudy saberá quando acertou e quando errou. Na Figura 8.5, ela acertou nos bits 0, 1, 2, 3, 4, 6, 8, 12 e 13. No entanto, ela sabe pela resposta de Alice na Figura 8.5(d) que só os bits 1, 3, 7, 8, 10, 11, 12 e 14 fazem parte da chave única. Em quatro desses bits (1, 3, 8 e 12), ela acertou seu palpite e captou o bit correto. Nos outros quatro (7, 10, 11 e 14), ela errou e não sabe qual bit foi transmitido. Desse modo, Bob sabe que a chave única começa com 01011001, a parir da Figura 8.5(e), mas tudo que Trudy tem é 01?1??0?, a partir da Figura 8.5(g).

É claro que Alice e Bob estão cientes de que Trudy talvez tenha captado parte de sua chave única, e assim gostariam de reduzir as informações que Trudy tem. Eles podem fazer isso executando uma transformação na chave. Por exemplo, poderiam dividir a chave única em blocos de 1024 bits e elevar ao quadrado cada uma para formar um número de 2048 bits, usando a concatenação desses números de 2048 bits como a chave única. Com seu conhecimento parcial do string de bits transmitido, Trudy não tem como gerar seu quadrado e, portanto, não tem nada. A transformação da chave única original em uma chave diferente que reduz o conhecimento de Trudy é chamada **amplificação da privacidade**. Na prática, são usadas transformações complexas em que todo bit de entrada depende de cada bit de saída em lugar da elevação ao quadrado.

Pobre Trudy. Ela não apenas não tem nenhuma idéia de qual é a chave única, mas sua presença não é mais secreta. Afinal, ela tem de retransmitir cada bit recebido para Bob, a fim de levá-lo a pensar que está se comunicando com Alice. Porém, o melhor que ela pode fazer é transmitir o qubit que recebeu, usando a mesma

polarização que empregou para recebê-lo, e durante cerca de metade do tempo ela estará errada, provocando muitos erros na chave única de Bob.

Quando finalmente começar a transmitir dados, Alice os codificará usando um pesado código de correção antecipada de erros. Do ponto de vista de Bob, um erro de 1 bit na chave única é o mesmo que um erro de transmissão de 1 bit. De qualquer modo, ele receberá o bit errado. Se houver correção antecipada de erros suficiente, ele poderá recuperar a mensagem original apesar de todos os erros, mas poderá contar com facilidade quantos erros foram corrigidos. Se esse número for muito maior que a taxa de erros esperada do equipamento, ele saberá que Trudy grampeou a linha e poderá agir de acordo (por exemplo, informando a Alice que ela deve mudar para um canal de rádio, chamar a polícia etc.). Se Trudy tivesse um meio de clonar um fóton, de forma que ela tivesse um fóton para inspecionar e um fóton idêntico para enviar a Bob, ela poderia evitar a detecção mas, no momento, não se conhece nenhum modo perfeito de clonar um fóton. No entanto, mesmo que Trudy pudesse clonar fótons, o valor da criptografia quântica para estabelecer chaves únicas não seria reduzido.

Embora a criptografia quântica opere sobre distâncias de até 60 km de fibra, o equipamento é complexo e dispendioso. Ainda assim, a idéia é promissora. Para obter mais informações sobre a criptografia quântica, consulte (Mullins, 2002).

[T3] 8.1.5 Dois princípios fundamentais da criptografia

Ainda estudaremos muitos sistemas criptográficos diferentes nas próximas páginas, mas é importante entender dois princípios básicos subjacentes a todos eles.

[T4] Redundância

O primeiro princípio é que todas as mensagens criptografadas devem conter

alguma redundância, ou seja, informações que não são necessárias para a compreensão da mensagem. Talvez um exemplo esclareça por que isso é necessário. Considere uma empresa de encomendas postais, a The Couch Potato (TCP), com 60.000 produtos. Pensando que estavam sendo muito eficientes, os programadores da TCP decidiram que as mensagens de encomendas deveriam consistir no nome do cliente com 16 bytes, seguido por um campo de dados de 3 bytes (um para a quantidade e 2 para o número do produto). Os 3 últimos bytes devem ser criptografados por meio de uma chave muito longa conhecida apenas pelo cliente e pela TCP.

Em princípio, essa estratégia pode parecer segura, e até certo ponto isso acontece, porque os intrusos passivos não podem descriptografar as mensagens. Infelizmente, há uma falha fatal que a torna inútil. Suponha que uma funcionária recém-demitida queira punir a TCP por despedi-la. Antes de sair da empresa, ela leva consigo parte da lista de clientes e passa a noite acordada criando um programa para gerar encomendas fictícias utilizando nomes de clientes verdadeiros. Como não tem a lista das chaves, ela simplesmente inclui números aleatórios nos três últimos bytes e envia centenas de encomendas para a TCP. Quando as mensagens chegam, o computador da TCP utiliza o nome do cliente para localizar a chave e descriptografar a mensagem. Infelizmente para a TCP, quase todas as mensagens de 3 bytes são válidas; portanto, o computador começa a imprimir as instruções de entrega. Apesar de parecer estranho um cliente encomendar 837 conjuntos de balanços para crianças, ou 540 caixas de areia, para o computador, o cliente pode estar planejando abrir uma cadeia de parques de diversões franqueados. Portanto, um intruso ativo (a ex-funcionária) pode causar muitos problemas, mesmo que não seja capaz de entender as mensagens que seu computador está gerando.

Esse problema pode ser resolvido através da inclusão de informações

redundantes em todas as mensagens. Por exemplo, se as mensagens de pedidos forem ampliadas para 12 bytes, os 9 primeiros deverão ser iguais a zero; assim, essa estratégia de ataque deixa de ser interessante, porque a ex-funcionária não é mais capaz de gerar um longo fluxo de mensagens válidas. A moral da história é que todas as mensagens devem conter informações redundantes suficientes para que os intrusos ativos sejam impedidos de transmitir dados inválidos que possam ser interpretados como uma mensagem válida.

No entanto, a inclusão de informações redundantes também facilita a ruptura de mensagens por parte dos criptoanalistas. Suponha que a empresa de encomenda postal seja muito competitiva e esteja na posição de principal concorrente da The Couch Potato. A Sofa Tuber adoraria saber quantas caixas de areia a TCP está vendendo. Portanto, a empresa resolve grampear a linha telefônica da TCP. No esquema original com mensagens de 3 bytes, a criptoanálise era praticamente impossível porque, após descobrir uma chave, o criptoanalista não era capaz de dizer se a mensagem estava correta. Afinal de contas, quase todas as mensagens são tecnicamente válidas. Com o novo esquema de 12 bytes, fica mais fácil para o criptoanalista distinguir uma mensagem válida de uma inválida. Desse modo, temos:

Princípio criptográfico 1: *as mensagens devem conter alguma redundância*

Em outras palavras, ao decifrar uma mensagem, o destinatário deve ser capaz de saber se ela é válida, simplesmente inspecionando-a e talvez executando uma computação simples. Essa redundância é necessária para impedir que intrusos ativos enviem lixo e enganem o receptor, fazendo-o descriptografar o lixo e agir sobre o "texto simples". No entanto, essa mesma redundância permite que os intrusos passivos entrem no sistema com maior facilidade; portanto, há uma zona de tensão nessa situação. Além disso, a redundância nunca deverá ser criada sob a forma de *n* zeros no início ou no fim de uma mensagem, pois a submissão

dessas mensagens a determinados algoritmos criptográficos proporciona resultados mais previsíveis, facilitando o trabalho do criptoanalista. Um polinômio de CRC é muito melhor que uma seqüência de valores 0, pois o receptor pode verificá-lo facilmente, mas ele irá gerar mais trabalho para o criptoanalista. Seria muito melhor usar um hash criptográfico, um conceito que exploraremos mais adiante.

Voltando à criptografia quântica por um momento, também podemos ver como a redundância desempenha um papel importante. Devido à interceptação dos fótons por Trudy, alguns bits na chave única de Bob estarão errados. Bob precisa de alguma redundância nas mensagens de entrada para descobrir os erros presentes. Uma forma muito rudimentar de redundância é repetir a mensagem duas vezes. Se as duas cópias não forem idênticas, Bob saberá que a fibra está muito ruidosa, ou que alguém está interferindo na transmissão. É claro que enviar tudo duas vezes é um exagero; um código de Hamming ou de Reed-Solomon é um modo mais eficiente de realizar a detecção e correção de erros. Porém, deve ficar claro que uma certa redundância é necessária para distinguir uma mensagem válida de uma mensagem inválida, em especial diante de um intruso ativo.

[T4] Atualidade

O segundo princípio criptográfico é tomar algumas medidas para assegurar que cada mensagem recebida possa ser confirmada como uma mensagem atual, isto é, enviada muito recentemente. Essa medida é necessária para impedir que intrusos ativos reutilizem mensagens antigas. Se tais medidas não fossem tomadas, nossa ex-funcionária poderia interceptar a linha telefônica da TCP e ficar simplesmente repetindo mensagens válidas já enviadas. Em outras palavras, essa idéia nos diz que:

Princípio criptográfico 2: *algum método é necessário para anular ataques de repetição*

Uma medida desse tipo seria incluir em cada mensagem um timbre de hora válido apenas por, digamos, 10 segundos. Em seguida, o receptor poderia manter as mensagens durante 10 segundos, a fim de comparar as mensagens recém-chegadas com as anteriores e filtrar duplicatas. As mensagens transmitidas há mais de 10 segundos poderiam ser descartadas, pois as repetições enviadas mais de 10 segundos depois da mensagem original serão rejeitadas por serem muito antigas. Outras medidas além dos timbres de hora serão discutidas mais adiante.

[T2] 8.2 Algoritmos de chave simétrica

Embora a criptografia moderna utilize as mesmas idéias básicas da criptografia tradicional (transposição e substituição), sua ênfase é diferente. Tradicionalmente, as pessoas que criam a criptografia têm utilizado algoritmos simples. Hoje em dia, acontece o inverso: o objetivo é tornar o algoritmo de criptografia tão complexo e emaranhado que, mesmo que o criptoanalista adquira enormes volumes de texto cifrado de sua própria escolha, sem a chave ele não seja capaz de captar qualquer sentido em tudo que conseguir.

A primeira classe de algoritmos de criptografia que estudaremos neste capítulo é a dos **algoritmos de chave simétrica**, porque utilizam a mesma chave para codificação e decodificação. A Figura 8.2 ilustra o uso de um algoritmo de chave simétrica. Em particular, vamos nos concentrar nas cifras de bloco, que obtêm um bloco de *n* bits de texto simples como entrada e o transformam usando a chave em um bloco de *n* bits de texto cifrado.

Os algoritmos criptográficos podem ser implementados em hardware (para se obter velocidade) ou em software (para se obter flexibilidade). Embora a maior parte de nosso tratamento esteja relacionado aos algoritmos e protocolos, que

são independentes da implementação real, algumas palavras sobre a construção de hardware criptográfico podem ser interessantes. As transposições e substituições podem ser implementadas com circuitos elétricos simples. A Figura 8.6(a) mostra um dispositivo, conhecido como **caixa P** (onde P significa permutação), usado para efetuar uma transposição em uma entrada de 8 bits. Se os 8 bits forem designados de cima para baixo como 01234567, a saída dessa caixa P específica será 36071245. Com uma fiação interna adequada, pode-se criar uma caixa P para executar qualquer transposição praticamente na velocidade da luz, pois nenhuma computação é envolvida, apenas a propagação e sinais. Esse projeto segue o princípio de Kerckhoff: o atacante sabe que o método geral é permutar os bits. O que ele não sabe é qual bit fica em cada posição, e isso é a chave.

[arte: ver original p. 737]

[Dísticos]

[1] Caixa P

(a)

[2] Caixa S

Decodificador: 3 para 8

Codificador: 8 para 3

(b)

[3] Cifra de produto

S_1 S_5 S_9

S_2 S_6 S_{10}

P_1 P_2 P_3 P_4

S_3 S_7 S_{11}

S_4 S_8 S_{12}

(c)

[F]Figura 8.6

[FL] Elementos básicos de cifras de produtos. (a) Caixa P. (b) Caixa S. (c) Produto

As substituições são realizadas por **caixas S**, como mostra a Figura 8.6(b). Nesse exemplo, é introduzido um texto simples de 3 bits, e a saída é um texto cifrado de 3 bits. A entrada de 3 bits seleciona uma das oito linhas de saída do primeiro estágio e a define como 1; todas as outras são iguais a 0. O segundo estágio é uma caixa P. O terceiro estágio codifica a linha selecionada novamente em binário. Com a fiação mostrada, se os oito números octais 01234567 fossem introduzidos um após o outro, a seqüência de saída seria 24506713. Em outras palavras, 0 foi substituído por 2, 1 foi substituído por 4 etc. Mais uma vez, com a fiação apropriada da caixa P dentro da caixa S, qualquer substituição pode ser realizada. Além disso, tal dispositivo pode ser construído em hardware e pode alcançar grande velocidade, pois os codificadores e os decodificadores têm apenas um ou dois retardos de porta (subnanossegundo) e o tempo de propagação pela caixa P pode ser menor que 1 picossegundo.

A capacidade real desses elementos básicos se torna aparente quando dispomos uma série inteira de caixas em cascata para formar uma **cifra de produto**, como mostra a Figura 8.6(c). Nesse exemplo, 12 linhas de entrada são transpostas (isto é, permutadas) pelo primeiro estágio (P_1). Teoricamente, seria possível fazer com que o segundo estágio fosse uma caixa S que mapeasse um número de 12 bits em outro número de 12 bits. No entanto, tal dispositivo necessitaria de 2^{12} = 4096 fios cruzados em seu estágio intermediário. Em vez disso, a entrada é dividida em quatro grupos de 3 bits, sendo que cada um deles é substituído de forma independente dos outros. Apesar de ser menos genérico, esse método ainda é mais eficiente. Através da inclusão de um número de estágios suficientemente grande na cifra de produto, a saída pode ser transformada em

uma função excessivamente complicada da entrada.

As cifras de produto que operam sobre entradas de *k* bits para produzir saídas de *k* bits são muito comuns. Em geral, o valor de *k* varia de 64 a 256. Uma implementação de hardware normalmente tem pelo menos 18 estágios físicos, em vez de apenas sete, como na Figura 8.6(c). Uma implementação de software é programada como um loop com pelo menos 8 iterações, cada uma executando substituições semelhantes às de caixas S em sub-blocos do bloco de dados de 64 bits a 256 bits, seguidas por uma permutação que mistura as saídas das caixas S. Com freqüência, existe uma permutação especial no início e também uma no fim. Na literatura, as repetições são chamadas **rodadas**.

[T3] 8.2.1 DES — Data Encryption Standard

Em janeiro de 1977, o governo dos Estados Unidos adotou uma cifra de produto desenvolvida pela IBM como seu padrão oficial para informações não confidenciais. A cifra, **DES (Data Encryption Standard — padrão de criptografia de dados)**, foi amplamente adotada pelo setor de informática para uso em produtos de segurança. Em sua forma original, ela já não é mais segura; no entanto, em uma forma modificada ela ainda é útil. Agora, amos explicar como o DES funciona.

Na Figura 8.7(a), é mostrado um esboço do DES. O texto simples é criptografado em blocos de 64 bits, produzindo 64 bits de texto cifrado. O algoritmo, parametrizado por uma chave de 56 bits, tem 19 estágios distintos. O primeiro deles é uma transposição independente da chave no texto simples de 64 bits. O último estágio é exatamente o inverso dessa transposição. O penúltimo estágio troca os 32 bits mais à esquerda pelos 32 bits mais à direita. Os 16 estados restantes são funcionalmente idênticos, mas são parametrizados por diferentes funções da chave. O algoritmo foi projetado para permitir que a decodificação

fosse feita com a mesma chave da codificação, uma propriedade necessária em qualquer algoritmo de chave simétrica. As etapas são simplesmente executadas na ordem inversa.

A operação desses estágios intermediários é ilustrada na Figura 8.7(b). Cada estágio utiliza duas entradas de 32 bits e produz duas saídas de 32 bits. A saída da esquerda é apenas uma cópia da saída da direita. A saída da direita é formada pelo resultado do OR exclusivo bit a bit aplicado à entrada da esquerda e a uma função da entrada da direita com a chave desse estágio, K_i. Toda a complexidade reside nessa função.

A função consiste em quatro etapas, executadas em seqüência. Primeiro, um número de 48 bits, E, é construído através da expansão do R_{i-1} de 32 bits, de acordo com uma regra fixa de transposição e duplicação. Em segundo lugar, E e K_i são submetidos a uma operação XOR. Em seguida, essa saída é particionada em oito grupos de 6 bits, sendo cada um deles entregue a uma caixa S diferente. Cada uma das 64 entradas possíveis para uma caixa S é mapeada em uma saída de 4 bits. Por fim, esses 8×4 bits passam por uma caixa P.

Em cada uma das 16 iterações, é utilizada uma chave diferente. Antes de se iniciar o algoritmo, é aplicada à chave uma transposição de 56 bits. Antes de cada iteração, a chave é particionada em duas unidades de 28 bits, sendo cada uma delas girada à esquerda um número de bits que depende do número da iteração. K_i é derivada dessa chave girada, pela aplicação de mais uma transposição de 56 bits sobre ela. Em cada rodada, um subconjunto de 48 bits dos 56 bits é extraído e permutado.

[arte: ver original p. 739]

[Dísticos]

[1] Chave de 56 bits

[2] Texto simples de 64 bits

Transposição inicial

Iteração 1

Iteração 2

.

.

.

Iteração 16

Troca (swap) de 32 bits

Transposição inversa

Texto cifrado de 64 bits

(a)

[3] L_{i-1} R_{i-1}

[ver símbolo]

32 bits 32 bits

L_i R_i

(b)

[F]Figura 8.5

[FL] O DES (Data Encryption Standard — padrão de criptografia de dados). (a) Esboço geral. (b) Detalhe de uma iteração. O sinal de adição dentro do círculo significa OR exclusivo (XOR)

Uma técnica às vezes utilizada para tornar o DES mais forte é chamada **@@@branqueamento**. Ela consiste em operação XOR entre uma chave aleatória de 64 bits e cada bloco de texto simples, antes de sua entrega ao DES e depois uma operação XOR ente uma segunda chave de 64 bits e o texto cifrado resultante, antes de sua transmissão. O branqueamento pode ser removido com facilidade pela execução das operações inversas (se o receptor tiver as duas chaves de

branqueamento). Tendo em vista que essa técnica acrescenta efetivamente mais bits ao tamanho da chave, ela torna uma pesquisa exaustiva do espaço de chaves muito mais demorada. Observe que a mesma chave de branqueamento é utilizada para cada bloco (isto é, só existe uma chave de branqueamento).

O DES esteve envolvido em controvérsias desde seu lançamento. Ele se baseia em uma cifra desenvolvida e patenteada pela IBM, cujo nome é Lucifer. A diferença é que a cifra da IBM utilizava uma chave de 128 bits, em vez de uma chave de 56 bits. Quando quis padronizar o uso de uma cifra para informações não confidenciais, o governo dos Estados Unidos "convidou" a IBM para "discutir" o problema com a NSA, o órgão do governo especializado em decifrar códigos, que é o maior empregador de matemáticos e criptólogos do mundo. A NSA é tão secreta que no setor existe a seguinte brincadeira com seu nome:

P: O que significa NSA?

R: No Such Agency (em português: não existe tal agência).

Na verdade, NSA significa National Security Agency.

Após essas discussões, a IBM reduziu a chave de 128 para 56 bits e decidiu manter em segredo o processo segundo o qual o DES foi projetado. Muita gente suspeitou de que o tamanho da chave foi reduzido para garantir que a NSA pudesse decifrar o DES, mas nenhuma organização com um orçamento menor foi de fazê-lo. Supostamente, o motivo para manter o projeto em segredo foi ocultar uma porta dos fundos (back door) que pudesse facilitar ainda mais a decifração do DES por parte da NSA. Quando um funcionário da NSA disse discretamente para o IEEE cancelaria uma conferência sobre criptografia, as pessoas ficaram ainda mais desconfortáveis com a situação. A NSA negou tudo.

Em 1977, dois pesquisadores de criptografia de Stanford, Diffie e Hellman (1977), projetaram uma máquina para decifrar o DES e estimaram que ela poderia ser montada por um custo de 20 milhões de dólares. Com base em um pequeno

trecho de texto simples e no texto cifrado correspondente, essa máquina poderia descobrir a chave através de uma pesquisa exaustiva do espaço de chaves de 2^{56} entradas em menos de 1 dia. Atualmente, essa máquina custaria bem menos de 1 milhão de dólares.

[T4] DES triplo

No início de 1979, a IBM percebeu que o tamanho da mensagem DES era muito pequeno e criou uma forma de aumentá-lo usando a criptografia tripla (Tuchman, 1979). O método escolhido, que desde então foi incorporado ao padrão internacional 8732, está ilustrado na Figura 8.8. Nesse caso, são usados três estágios e duas chaves. No primeiro estágio, o texto simples é criptografado com K_1 da maneira usual do DES. No segundo estágio, o DES é executado no modo de descriptografia, com o uso de K_2 como chave. Por fim, outra criptografia é feita com K_1.

[arte: ver original p. 741]

[Dísticos]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.8

[FL] (a) Criptografia tripla usando o DES. (b) Descriptografia

Esse projeto levanta duas questões. Primeiro, por que são utilizadas apenas duas chaves em vez de três? Segundo, por que foi usado **EDE (Encrypt Decrypt Encrypt)**, em vez de **EEE (Encrypt Encrypt Encrypt)**? São utilizadas duas chaves porque até mesmo os criptógrafos mais paranóicos concordam que 112 bits serão suficientes para aplicações comerciais durante um bom tempo. (E, entre os

criptógrafos, a paranóia é considerada um recurso, não um bug.) O uso de 168 bits só criaria overhead desnecessário de gerenciar e transportar outra chave, com pouco ganho real.

O motivo para criptografar, descriptografar e criptografar mais uma vez é a compatibilidade retroativa com os sistemas DES de chave única existentes. Tanto as funções de criptografia quanto as de descriptografia são mapeamentos entre conjuntos de números de 64 bits. Do ponto de vista da criptografia, os dois mapeamentos são igualmente fortes. No entanto, ao usar EDE em vez de EEE, um computador que utiliza a criptografia tripla pode se comunicar com outro que utiliza a criptografia apenas definindo $K_1 = K_2$. Essa propriedade permite que a criptografia tripla seja ajustada gradualmente, o que não interessa aos criptógrafos acadêmicos, mas que é de grande importância para a IBM e seus clientes.

[T3] 8.2.2 AES — Advanced Encryption Standard

À medida que o DES começou a se aproximar do fim de sua vida útil, mesmo com o DES triplo, o **NIST (National Institute of Standards and Technology)**, o órgão do departamento de comércio dos Estados Unidos encarregado de aprovar padrões para o Governo Federal dos Estados Unidos, decidiu que o governo precisava de um novo padrão criptográfico para uso não confidencial. O NIST estava ciente de toda a controvérsia que cercava o DES e sabia muito bem que, se anunciasse um novo padrão, todas as pessoas que soubessem algo sobre criptografia concluiriam automaticamente que a NSA havia criado uma porta dos fundos no DES, e assim a NSA poderia ler tudo que fosse criptografado com ele. Sob essas condições, talvez ninguém utilizasse o padrão e seria mais provável que ele desaparecesse.

Dessa forma, o NIST adotou uma estratégia diferente e surpreendente para um

órgão do governo: patrocinou um concurso de criptografia. Em janeiro de 1997, pesquisadores do mundo inteiro foram convidados a submeter propostas para um novo padrão, a ser chamado **AES (Advanced Encryption Standard)**. As regras do concurso eram:

1. O algoritmo teria de ser uma cifra de bloco simétrica.
2. Todo o projeto teria de ser público.
3. Deveriam ser admitidos tamanhos de chaves iguais a 128, 192 e 256 bits.
4. Teriam de ser possíveis implementações de software e de hardware.
5. O algoritmo teria de ser público ou licenciado em termos não discriminatórios.

Foram feitas quinze propostas sérias e foram organizadas conferências públicas nas quais essas propostas eram apresentadas, e os participantes eram encorajados ativamente a encontrar falhas em todas elas. Em agosto de 1998, o NIST selecionou cinco finalistas, baseado principalmente em seus requisitos de segurança, eficiência, simplicidade, flexibilidade e memória (importante para sistemas incorporados). Foram realizadas outras conferências e mais tentativas de encontrar falhas nos algoritmos. Na última conferência, houve uma votação sem compromisso. Os finalistas e suas pontuações foram:

1. Rijndael (de Joan Daemen e Vincent Rijmen, 86 votos).
2. Serpent (de Ross Anderson, Eli Biham e Lars Knudsen, 59 votos).
3. Twofish (de uma equipe liderada por Bruce Schneier, 31 votos).
4. RC6 (da RSA Laboratories, 23 votos).
5. MARS (da IBM, 13 votos).

Em outubro de 2000, o NIST anunciou que também votou no Rijndael e, em novembro de 2001, o Rijndael se tornou um padrão do Governo dos Estados Unidos publicado como Federal Information Processing Standard FIPS 197. Devido à extraordinária abertura da competição, às propriedades técnicas do Rijndael e ao fato de que a equipe premiada consistia em dois jovens criptógrafos belgas

(com pouca probabilidade de terem criado uma porta dos fundos só para agradar a NSA), esperava-se que o Rijndael se tornasse o padrão criptográfico dominante no mundo por pelo menos uma década. O nome Rijndael deriva dos sobrenomes dos autores: Rijmen + Daemen.

O Rijndael admite tamanhos de chaves e tamanhos de blocos desde 128 bits até 256 bits em intervalos de 32 bits. O comprimento da chave e o do bloco podem ser escolhidos independentemente. Porém, o AES especifica que o tamanho do bloco deve ser 128 bits e o comprimento da chave deve ser 128, 192 ou 256 bits. É pouco provável que alguém utilize chaves de 192 bits; assim, de fato, o AES tem duas variantes: um bloco de 128 bits com uma chave de 128 bits e um bloco de 128 bits com uma chave de 256 bits.

Em nosso tratamento do algoritmo apresentado a seguir, examinaremos apenas o caso de 128/128, porque é provável que esse se torne a norma comercial. Uma chave de 128 bits oferece um espaço de chaves de $2^{128} \approx 3 \times 10^{38}$ chaves. Ainda que o NSA consiga construir uma máquina com 1 bilhão de processadores paralelos, cada um capaz de avaliar uma chave por picossegundo, tal máquina levaria cerca de 10^{10} anos para pesquisar o espaço de chaves. Nessa época, o sol já terá explodido e as pessoas que sobreviverem terão de ler os resultados a luz de vela.

[T4] Rijndael

Sob uma perspectiva matemática, o Rijndael se baseia na teoria de campo de Galois, o que proporciona ao algoritmo algumas propriedades de segurança demonstráveis. Porém, ele também pode ser visto como código em C, sem a necessidade de entrarmos nos detalhes matemáticos.

Como o DES, o Rijndael utiliza substituição e permutações, e também emprega várias rodadas. O número de rodadas depende do tamanho da chave e do

tamanho do bloco, sendo 10 para chaves de 128 bits com blocos de 128 bits, passando para 14 no caso da maior chave ou do maior bloco. No entanto, diferente do DES, todas as operações envolvem bytes inteiros, a fim de permitir implementações eficientes, tanto em hardware quanto em software. Um esboço do código é apresentado na Figura 8.9.

[arte: ver original da p. 743]

[TD]

```
#define LENGTH 16                                   /* número de bytes em bloco
de dados ou chave */
#define NROWS 4                                     /* número de linhas em
estado */
#define NCOLS 4                                     /* número de colunas em
estado */
#define ROUNDS 10                                   /* número de iterações */
typedef unsigned char byte;                         /* inteiro de 8 bits sem sinal
*/

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                            /* índice de loop */
  byte state[NROWS][NCOLS];                         /* estado atual */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];    /* chaves de rodadas */
  expand_key(key, rk);                              /* cria as chaves de rodadas
*/
  copy_plaintext_to_state(state, plaintext);        /* inicializa estado atual*/
  xor_roundkey_into_state(state, rk[0]);            /* efetua XOR de chave em
estado */
```

```
  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                          /* aplica caixa S a cada byte
*/
      rotate_rows(state);                         /* gira linha i por i bytes */
      if (r < ROUNDS) mix_columns(state);         /* função mix */
      xor_roundkey_into_state(state, rk[r]);      /* efetua XOR de chave em
estado */
  }
  copy_state_to_ciphertext(ciphertext, state);    /* retorna resultado */
} [TN]
```

[F]Figura 8.9

[FL] Um esboço do Rijndael

A função *rijndael* tem três parâmetros. Esses parâmetros são: *plaintext*, um array de 16 bytes contendo os dados de entrada, *ciphertext*, um array de 16 bytes no qual será retornada a saída cifrada e *key*, a chave de 16 bytes. Durante o cálculo, o estado atual dos dados é mantido no array de bytes *state*, cujo tamanho é *NROWS* × *NCOLS*. No caso de blocos de 128 bits, esse array tem 4 × 4 bytes. Em 16 bytes, é possível armazenar todo o bloco de dados de 128 bits.

O array *state* é inicializado como o texto simples e modificado por cada etapa da computação. Em algumas etapas, é executada a substituição byte a byte. Em outras etapas, os bytes são permutados dentro do array. Outras transformações também são usadas. No final, o conteúdo do array *state* é retornado como texto cifrado.

O código começa expandindo a chave em 11 arrays do mesmo tamanho que o estado. Eles são armazenados em *rk*, um array de structs, cada uma contendo um array de estado. Um desses arrays será utilizado no início do cálculo, e os outros

10 serão usados durante as 10 rodadas, um por rodada. O cálculo das chaves de rodadas a partir da chave de criptografia é muito complicado para ser examinado aqui. Basta saber que as chaves de rodadas são produzidas pela rotação e pela operação XOR repetida de vários grupos de bits da chave. Para ver todos os detalhes, consulte (Daemen e Rijmen, 2002).

A próxima etapa é copiar o texto simples no array *state* de forma que ele possa ser processado durante as rodadas. O texto é copiado em ordem de colunas, com os quatro primeiros bytes na coluna 0, os quatro bytes seguintes na coluna 1 e assim por diante. Tanto as colunas quanto as linhas são numeradas a partir de 0, embora as rodadas se iniciem em 1. Essa configuração inicial dos 12 arrays de bytes com tamanho 4 × 4 é ilustrada na Figura 8.10.

[arte: ver original p. 744]

[Dísticos]

[1] Texto simples de 128 bits

[2] Chave de criptografia de 128 bits

[3] state rk[0] rk[1] rk[2] rk[3] rk[4] rk[5] rk[6] rk[7] rk[8] rk[9] rk[10]

[4] Chaves de rodadas

[F]Figura 8.10

[FL] Criação dos arrays *state* e *rk*

Há mais uma etapa antes do início da principal computação: *rk*[0] é submetido a uma operação XOR em *state* byte por byte. Em outras palavras, cada um dos 16 bytes em *state* é substituído pelo XOR do próprio valor com o byte correspondente em *rk*[0].

Agora chegou a hora da atração principal. O loop executa 10 repetições, uma por rodada, transformando *state* em cada repetição. O conteúdo de cada redonda

consiste em quatro etapas. A etapa 1 efetua uma substituição byte a byte em *state*. Por sua vez, cada byte é usado como um índice para uma caixa S, a fim de substituir seu valor pelo conteúdo dessa entrada da caixa S. Essa etapa é uma cifra de substituição monoalfabética. Diferente do DES, que tem diversas caixas S, o Rijndael tem apenas uma caixa S.

A etapa 2 gira cada uma das quatro linhas para a esquerda. A linha 0 é girada 0 bytes (isto é, não se altera), a linha 1 é girada 1 byte, a linha 2 é girada 2 bytes e a linha 3 é girada 3 bytes. Essa etapa difunde o conteúdo dos dados atuais em torno do bloco, de modo análogo às permutações da Figura 8.6.

A etapa 3 mistura cada coluna independentemente das outras. A mistura é realizada com o uso da multiplicação de matrizes, na qual a nova coluna é o produto da coluna antiga por uma matriz constante, sendo a multiplicação feita com o campo finito de Galois, $GF(2^8)$. Embora isso possa parecer complicado, existe um algoritmo que permite calcular cada elemento da nova coluna usando duas pesquisas de tabelas e três operações XOR (Daemen e Rijmen, 2002, Apêndice E).

Finalmente, a etapa 4 efetua o XOR da chave correspondente a essa rodada no array *state*.

Tendo em vista que cada etapa é reversível, a decodificação pode ser feita simplesmente executando-se o algoritmo no sentido inverso. Porém, também existe um artifício, pelo qual a decodificação pode ser realizada executando-se o algoritmo de criptografia com a utilização de tabelas diferentes.

O algoritmo foi projetado não só por segurança, mas também para aumentar a velocidade. Uma boa implementação de software em uma máquina de 2 GHz deve ser capaz de alcançar uma taxa de criptografia de 700 Mbps, que é rápida o suficiente para codificar mais de 100 vídeos MPEG-2 em tempo real. As implementações de hardware são ainda mais rápidas.

[T3] 8.2.3 Modos de cifra

Apesar de toda essa complexidade, o AES (ou o DES, ou ainda qualquer cifra de bloco) é basicamente uma cifra de substituição monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para AES, e caracteres de 64 bits para DES). Sempre que o mesmo bloco de texto simples chega ao front end, o mesmo bloco de texto cifrado sai pelo back end. Se codificar o texto simples *abcdefgh* 100 vezes com a mesma chave DES, você obterá o mesmo texto cifrado 100 vezes. Um intruso pode explorar essa propriedade para ajudar a subverter a cifra.

[T4] O modo Electronic Code Book

Para ver como essa propriedade das cifras de substituição monoalfabética podem ser usadas para anular parcialmente a cifra, usaremos o DES (triplo), por ser mais fácil representar blocos de 64 bits que blocos de 128 bits, mas o AES tem exatamente o mesmo problema. A maneira direta de usar o DES para codificar um longo fragmento de texto simples é dividi-lo em blocos consecutivos de 8 bytes (64 bits) e codificá-los uns após outros com a mesma chave. O último fragmento de texto simples é completado até 64 bits, se necessário. Essa técnica é conhecida como **modo ECB (modo Electronic Code Book)** em analogia aos antigos livros de código em que cada palavra de texto simples era listada, seguida por seu texto cifrado (em geral, um número decimal de cinco dígitos).

Na Figura 8.11, temos o início de um arquivo de computador listando as gratificações anuais que uma empresa decidiu oferecer a seus funcionários. Esse arquivo consiste em registros de 32 bytes consecutivos, um por funcionário, no formato mostrado: 16 bytes para o nome, 8 bytes para o cargo e 8 bytes para a gratificação. Cada um dos dezesseis blocos de 8 bytes (numerados de 0 até 15) é codificado pelo DES (triplo).

[arte: ver original p. 746]

[Dísticos]

[1] Nome Posição Gratificação

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.11

[FL] O texto simples de um arquivo codificado como 16 blocos DES

Leslie acabou de ter uma briga com o chefe e sabe que não deve esperar uma grande gratificação. Em contraste, Kim é a favorito do chefe e todo mundo sabe disso. Leslie pode acessar o arquivo após a codificação, mas antes do arquivo ser enviado ao banco. Leslie pode retificar essa situação injusta, tendo apenas o arquivo criptografado?

Não há problema. Leslie só precisa fazer uma cópia do 12° bloco de texto cifrado (que contém a gratificação de Kim) e usá-lo para substituir o 4° bloco de texto cifrado (que contém a gratificação de Leslie). Mesmo sem saber o que contém o 12° bloco, Leslie pode esperar ter um Natal muito mais feliz este ano. (Copiar o 8° bloco de texto cifrado também é uma possibilidade, mas a probabilidade de ser descoberto é maior; além disso, Leslie não é uma pessoa gananciosa.)

[T4] Modo de encadeamento de blocos de cifras

Para contrariar esse tipo de ataque, todos as cifras de blocos podem ser encadeadas de várias maneiras, para que a substituição de um bloco como a que Leslie fez transforme o texto simples decodificado em lixo, a partir do bloco substituído. Uma forma de encadeamento é o **encadeamento de blocos de cifras**. Nesse método, mostrado na Figura 8.12, cada bloco de texto simples é

submetido a uma operação XOR com o bloco de texto cifrado anterior, antes de ser codificado. Conseqüentemente, o mesmo bloco de texto simples não é mais mapeado para o mesmo bloco de texto cifrado, e a criptografia não é mais uma grande cifra de substituição monoalfabética. O primeiro bloco é submetido a uma operação XOR com um **IV (Initialization Vector — vetor de inicialização)**, escolhido ao acaso, que é transmitido (em texto simples) juntamente com o texto cifrado.

Podemos ver como funciona o modo de encadeamento de blocos de cifras examinando o exemplo da Figura 8.12. Começamos calculando $C_0 = E(P_0 \text{ XOR } IV)$. Em seguida, calculamos $C_1 = E(P_1 \text{ XOR } C_0)$ e assim por diante. A decodificação também utiliza XOR para inverter o processo, com $P_0 = IV \text{ XOR } D(C_0)$ e assim por diante. Observe que a criptografia de bloco i é uma função de todo o texto simples contidos nos blocos 0 a $i - 1$, e assim o mesmo texto simples gera um texto cifrado diferente, dependendo de onde ele ocorre. Uma transformação do tipo que Leslie fez resultará em texto sem sentido para dois blocos a partir do campo de gratificação de Leslie. Para um funcionário de segurança astuto, essa peculiaridade pode sugerir onde iniciar a investigação legal.

[arte: ver original p. 747]

[Dísticos]

[1] Chave

[2] Caixa de criptografia

[3] Chave

[4] Caixa de descriptografia

[5] OR exclusivo

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.12

[FL] Encadeamento e blocos de cifras. (a) Codificação. (b) Decodificação

O encadeamento de blocos de cifras também tem uma vantagem: o mesmo bloco de texto simples não resultará no mesmo bloco de texto cifrado. Assim, a criptoanálise será difícil. De fato, essa é a principal razão de seu uso.

[T4] Modo de feedback de cifra

No entanto, o encadeamento de blocos de cifras tem a desvantagem de exigir a chegada de um bloco de 64 bits inteiro para poder iniciar a decodificação. Quando é utilizado em terminais interativos, nos quais as pessoas podem digitar linhas com menos de oito caracteres e parar à espera de uma resposta, esse modo é inadequado. No caso da codificação byte a byte, é usado o **modo de feedback de cifra**, empregando o DES (triplo), como mostra a Figura 8.13. Para o AES, a idéia é exatamente a mesma, sendo usado um registrador de deslocamento de 128 bits. Na figura, o estado da máquina de criptografia é mostrado após os bytes 0 a 9 terem sido codificados e enviados. Ao chegar o byte 10 do texto simples, conforme ilustra a Figura 8.13(a), o algoritmo DES opera sobre o registrador de deslocamento de 64 bits para gerar um texto cifrado de 64 bits. O byte mais à esquerda desse texto cifrado é extraído e submetido a uma operação XOR com P_{10}. Depois, esse byte é encaminhado à linha de transmissão. Além disso, o registrador de deslocamento (shift register) é deslocado 8 bits à esquerda, fazendo C_2 ficar fora da extremidade esquerda, e C_{10} é inserido na posição que acabou de ficar vaga na extremidade direita, logo depois de C_9. Observe que o conteúdo do registrador de deslocamento depende de todo o histórico anterior do texto simples; assim, um padrão que se repetir várias vezes no texto simples será criptografado de maneira diferente do texto

cifrado a cada repetição. Como ocorre no encadeamento de blocos de cifras, é necessário um vetor de inicialização para dar início ao processo.

A decodificação com o modo de feedback de cifra funciona exatamente como a codificação. Em particular, o conteúdo do registrador de deslocamento é *codificado* e não *decodificado*, e assim o byte selecionado que é submetido à operação XOR com C_{10} para se obter P_{10} é o mesmo que sofreu a operação XOR com P_{10} para gerar C_{10} na primeira vez. Desde que os dois registradores de deslocamento permaneçam idênticos, a decodificação funcionar corretamente. Ela é ilustrada na Figura 8.13(b).

[arte: ver original p. 748]

[Dísticos]

[1] Registrador de deslocamento de 64 bits Registrador de deslocamento de 64 bits

C_2 C_3 C_4 C_5 C_6 C_7 C_9 C_2 C_3 C_4 C_5 C_6 C_7 C_9

[2] Chave E Caixa de criptografia C_{10}

[3] Chave E Caixa de criptografia C_{10}

[4] Seleciona o byte mais à esquerda

[5] Seleciona o byte mais à esquerda

[6]P_{10} + C_{10}

XOR

(a)

[7] C_{10} + P_{10}

(b)

[F]Figura 8.13

[FL] Modo de feedback de cifra. (a) Codificação. (b) Decodificação

O modo de feedback de cifra apresenta um problema: se um bit do texto cifrado

for invertido acidentalmente durante a transmissão, os 8 bytes decodificados enquanto o byte defeituoso estiver no registrador de deslocamento serão danificados. Depois que o byte defeituoso é empurrado para fora do registrador de deslocamento, o texto simples correto será gerado mais uma vez. Desse modo, os efeitos de um único bit invertido são relativamente localizados e não arruinam o restante da mensagem, mas arruinam uma quantidade de bits igual à largura do registrador de deslocamento.

[T4] Modo de cifra de fluxo

Apesar disso, existem aplicações em que um erro de transmissão de 1 bit alterando 64 bits de texto simples provoca um impacto grande demais. Para essas aplicações, há uma quarta opção, o **modo de cifra de fluxo**. Ele funciona codificando um vetor de inicialização, com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para se obter um segundo bloco de saída. Em seguida, esse bloco é codificado para se obter um terceiro bloco e assim por diante. A seqüência (arbitrariamente grande) de blocos de saída, chamada **fluxo de chaves**, é tratada como uma chave única e submetida a uma operação XOR com o texto simples para se obter o texto cifrado, como mostra a Figura 8.14(a). Observe que o vetor de inicialização só é usado na primeira etapa. Depois disso, a saída é codificada. Observe também que o fluxo de chaves é independente dos dados, e portanto pode ser calculado com antecedência, se necessário, e é completamente insensível a erros de transmissão. A decodificação é mostrada na Figura 8.14(b).

A decodificação ocorre gerando-se o mesmo fluxo de chaves no lado receptor. Como o fluxo de chaves só depende do vetor de inicialização e da chave, ele não é afetado por erros de transmissão no texto cifrado. Desse modo, um erro de 1 bit no texto cifrado transmitido gera apenas um erro de 1 bit no texto simples

decodificado.

[arte: ver original p. 749]

[Dísticos]

[1] IV

Caixa de criptografia

Chave E

Fluxo de chaves

Texto simples Texto cifrado

(a)

[2] IV

Caixa de criptografia

Chave E

Fluxo de chaves

Texto cifrado Texto simples

(b)

[F]Figura 8.14

[FL] Uma cifra de fluxo. (a) Codificação. (b) Decodificação

É essencial nunca utilizar o mesmo par (chave, IV) duas vezes com uma cifra de fluxo, porque isso irá gerar o mesmo fluxo de chaves o tempo todo. O uso de um mesmo fluxo de chaves duas vezes expõe o texto cifrado a um **ataque de reutilização de fluxo de chaves.** Imagine que o bloco de texto simples P_0 seja codificado com o fluxo de chaves para se obter P_0 XOR K_0. Mais tarde, um segundo bloco de texto simples Q_0 é codificado com o mesmo fluxo de chaves para se obter Q_0 XOR K_0. Um intruso que capturar ambos os blocos de texto cifrado poderá simplesmente efetuar uma operação XOR dos dois juntos para obter P_0 XOR Q_0, o que eliminará a chave. Agora, o intruso tem o XOR dos dois

blocos de texto simples. Se um deles for conhecido ou puder ser encontrado, o outro também poderá ser encontrado. Em todo caso, o XOR de dois fluxos de texto simples poderá ser atacado com a utilização de propriedades estatísticas da mensagem. Por exemplo, no caso de texto em inglês, o caractere mais comum no fluxo provavelmente será o XOR de dois espaços, seguido pelo XOR de espaço e da letra "e" etc. Em resumo, equipado com o XOR de dois textos simples, o criptoanalista tem uma excelente chance de deduzi-los.

[T4] Modo de contador

Um problema apresentado por todos os modos, com exceção do modo de livro de código eletrônico, é a impossibilidade de conseguir acesso aleatório a dados codificados. Por exemplo, suponha que um arquivo seja transmitido por uma rede e depois armazenado em disco em forma codificada. Isso poderia ser um meio razoável de operação, se o computador receptor fosse um notebook que pudesse ser roubado. Armazenar todos os arquivos críticos em forma codificada reduz muito os danos causados pelo vazamento de informações secretas na eventualidade do computador cair em mãos erradas.

Porém, com freqüência os arquivos de disco são acessados em ordem não seqüencial, especialmente arquivos de bancos de dados. No caso de um arquivo codificado pela utilização do encadeamento de blocos de cifras, o acesso a um bloco aleatório exige primeiro a decodificação de todos os blocos situados à frente dele, uma proposta dispendiosa. Por essa razão, foi criado mais um modo, o **modo de contador**, como ilustra a Figura 8.15. Aqui, o texto simples não é codificado diretamente. Em vez disso, o vetor de inicialização somado a uma constante é codificado, e o texto cifrado resultante é submetido a um XOR com o texto simples. Aumentar o vetor de inicialização em 1 unidade a cada novo bloco, facilita a decodificação de um bloco em qualquer lugar no arquivo sem que

primeiro seja preciso decodificar todos os seus predecessores.

[arte: ver original p. 750]

[Dísticos]

[1] Chave

[2] Caixa de codificação

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.15

[FL] Codificação com a utilização do modo de contador

Embora seja útil, o modo de contador tem um ponto fraco que vale a pena assinalar. Suponha que a mesma chave *K* seja usada novamente no futuro (com um texto simples diferente, mas com o mesmo IV) e um atacante adquira todo o texto cifrado de ambas as execuções. O fluxos de chaves são nos dois casos, expondo a cifra a um ataque de reutilização de fluxo de chaves do mesmo tipo que vimos no caso de cifras de fluxo. O criptoanalista tem de efetuar a operação XOR dos dois textos cifrados juntos, a fim de eliminar toda a proteção criptográfica e simplesmente obter o XOR dos textos simples. Essa debilidade não significa que o modo de contador é má idéia. Ela simplesmente quer dizer que tanto as chaves quanto os vetores de inicialização devem ser escolhidos de forma independente e ao acaso. Ainda que a mesma chave seja utilizada duas vezes por acidente, se o IV for diferente em cada utilização, o texto simples estará seguro.

[T3] 8.2.4 Outras cifras

DES e Rijndael são os algoritmos criptográficos de chave simétrica mais conhecidos. Porém, vale a pena mencionar que foram criados várias outras cifras

de chave simétrica. Algumas delas estão incorporadas a vários produtos. A Figura

8.16 mostra algumas dentre as cifras de chave simétrica mais comuns.

[arte: ver original p. 751]

[T]Tabela

Cifra

Blowfish

DES

IDEA

RC4

RC5

Rijndael

Serpent

DES triplo

Twofish

Autor

Bruce Schneier

IBM

Massey e Xuejia

Ronald Rivest

Ronald Rivest

Daemen e Rijmen

Anderson, Biham, Knudsen

IBM

Bruce Schneier

Comprimento da chave

1 a 448 bits

56 bits

128 bits

1 a 2048 bits

128 a 256 bits

128 a 256 bits

128 a 256 bits

168 bits

128 a 256 bits

Comentários

Velho e lento

Muito fraco para usar agora

Bom, mas patenteado

Atenção: algumas chaves são fracas

Bom, mas patenteado

Melhor escolha

Muito forte

Segunda melhor escolha

Muito forte; amplamente utilizado

[F]Figura 8.16

[FL] Alguns algoritmos criptográficos de chave simétrica comuns

[T3] 8.2.5 Criptoanálise

Antes de deixar o assunto de criptografia de chave simétrica, vale a pena mencionar pelo menos quatro desenvolvimentos em criptoanálise. O primeiro desenvolvimento é a **criptoanálise diferencial** (Biham e Shamir, 1993). Essa técnica pode ser usada para atacar qualquer cifra de bloco. Ela funciona a partir de um par de blocos e texto simples que diferem apenas por um pequeno número de bits e pela observação cuidadosa do que acontece em cada iteração

interna à medida que a codificação prossegue. Em muitos casos, alguns padrões de bits são muito mais comuns que outros, e essa observação leva a um ataque probabilístico.

O segundo desenvolvimento que vale a pena notas é a **criptoanálise linear** (Matsui, 1994). Ela pode romper o DES com apenas 2^{46} textos simples conhecidos. Essa técnica funciona efetuando o XOR de certos bits no texto simples e no texto cifrado, e examinando o resultado em busca de padrões. Quando isso é feito repetidamente, metade dos bits deve ter o valor 0 e metade deve ter o valor 1. Porém, com freqüência as cifras introduzem uma inclinação em um sentido ou no outro, e essa inclinação, embora pequena, pode ser explorada para reduzir o fator de trabalho. Para ver os detalhes, consulte o ensaio de Matsui.

O terceiro desenvolvimento é o uso da análise do consumo de energia elétrica para encontrar chaves secretas. Em geral, os computadores utilizam 3 volts para representar um 1 bit e 0 volts para representar um bit 0. Desse modo, o processamento de um bit 1 exige mais energia elétrica que o processamento de um 0. Se um algoritmo criptográfico consistir em um loop no qual os bits da chave são processados em ordem, um atacante que substituir o clock principal de *n* GHz por um clock lento (por exemplo, 100 Hz) e prender pinças dentadas (pinças jacaré) nos pinos de energia da CPU e de terra poderá monitorar com precisão a energia consumida por cada instrução de máquina. A partir desses dados, será surpreendentemente fácil deduzir a chave. Esse tipo de criptoanálise só pode ser anulado por codificação cuidadosa do algoritmo em linguagem assembly para ter certeza de que o consumo de energia será independente da chave e também independente de todas as chaves de rodadas individuais.

O quarto desenvolvimento é a análise de sincronismo. Os algoritmos criptográficos estão repletos de instruções [TD]if[TN] que testam bits nas chaves

de rodadas. Se as partes [TD]then[TN] e [TD]else[TN] demorarem períodos de tempo diferentes, tornando mais lento o clock e verificando quanto tempo demoram diversas etapas, talvez seja possível deduzir as chaves de rodadas. Uma vez que todas as chaves de rodadas sejam conhecidas, em geral será possível calcular a chave original. A análise da energia e da sincronização também podem ser empregadas simultaneamente para facilitar o trabalho. Embora a análise da energia e da sincronização possam parecer exóticas, na realidade são técnicas eficientes que podem romper qualquer cifra não projetada de forma específica para resistir a elas.

[T2] 8.3 Algoritmos de chave pública

Historicamente, o problema da distribuição de chaves sempre foi o elo mais fraco da maioria dos sistemas de criptografia. Independente de quanto um sistema de criptografia fosse sólido, se um intruso conseguisse roubar a chave, o sistema acabava sendo inútil. Como todos os criptólogos sempre presumem que a chave de criptografia e a chave de descriptografia são iguais (ou facilmente derivadas uma da outra) e que a chave é distribuída a todos os usuários do sistema, tinha-se a impressão de que havia um problema inerente ao sistema: as chaves tinham de ser protegidas contra roubo, mas também tinham de ser distribuídas; portanto, elas não podiam ser simplesmente trancadas na caixa-forte de um banco.

Em 1976, dois pesquisadores da University of Stanford, Diffie e Hellman (1976), propuseram um sistema de criptografia radicalmente novo, no qual as chaves de criptografia e de descriptografia eram diferentes, e a chave de descriptografia não podia ser derivada da chave de criptografia. Em sua proposta, o algoritmo de criptografia (chaveado) *E* e o algoritmo de descriptografia (chaveado) *D* tinham de atender a três requisitos, que podem ser declarados da seguinte forma:

1. $D(E(P)) = P$.

2. É extremamente difícil deduzir D a partir de E.

3. E não pode ser decifrado por um ataque de texto simples escolhido.

O primeiro requisito diz que, se aplicarmos D a uma mensagem criptografada, $E(P)$, obteremos outra vez a mensagem de texto simples original P. Sem essa propriedade, o destinatário legítimo não poderia decodificar o texto cifrado. O segundo é auto-explicativo. O terceiro é necessário porque, como veremos em um minuto, os intrusos podem experimentar o algoritmo até se cansarem. Sob essas condições, não há razão para a chave criptográfica não se tornar pública.

O método funciona da seguinte forma: uma pessoa, digamos Alice, desejando receber mensagens secretas, primeiro cria dois algoritmos que atendem aos requisitos anteriores. O algoritmo de criptografia e a chave de Alice se tornam públicos, daí o nome **criptografia de chave pública**. Por exemplo, Alice poderia colocar sua chave pública na home page que ela tem na Web. Usaremos a notação E_A para indicar o algoritmo de criptografia parametrizado pela chave pública de Alice. De modo semelhante, o algoritmo de descriptografia (secreto) parametrizado pela chave privada de Alice é D_A. Bob faz o mesmo, publicando E_B, mas mantendo secreta a chave D_B.

Agora vamos ver se podemos resolver o problema de estabelecer um canal seguro entre Alice e Bob, que nunca haviam tido um contato anterior. Supondo que tanto a chave de criptografia de Alice, E_A, quanto a chave de criptografia de Bob, E_B, estejam em arquivos de leitura pública. Agora, Alice pega sua primeira mensagem P, calcula $E_B(P)$ e a envia para Bob. Em seguida, Bob a descriptografa aplicando sua chave secreta D_B [ou seja, ele calcula $D_B(E_B(P)) = P$]. Ninguém mais pode ler a mensagem criptografada $E_B(P)$, porque o sistema de criptografia é considerado sólido e porque é muito difícil derivar D_B da chave E_B publicamente conhecida. Para enviar uma resposta R, Bob transmite $E_A(R)$. Agora, Alice e Bob podem se

comunicar com segurança.

Talvez seja interessante fazer uma observação sobre a terminologia. A criptografia de chave pública exige que cada usuário tenha duas chaves: uma chave pública, usada pelo mundo inteiro para criptografar as mensagens a serem enviadas para esse usuário, e uma chave privada, que o usuário utiliza para descriptografar mensagens. Faremos referência a essas chaves como *chave pública* e *chave privada*, respectivamente, e vamos distingui-las das chaves *secretas* (também chamadas chaves simétricas) usadas na criptografia de chave simétrica convencional.

[T3] 8.3.1 RSA

O único problema é que temos de encontrar algoritmos que realmente satisfaçam a todos os três requisitos. Devido às vantagens potenciais da criptografia de chave pública, muitos pesquisadores estão se dedicando integralmente a seu estudo, e alguns algoritmos já foram publicados. Um método muito interessante foi descoberto por um grupo de pesquisadores do MIT (Rivest *et al.*, 1978) e é conhecido pelas iniciais dos três estudiosos que o criaram (Rivest, Shamir, Adleman): **RSA.** Ele sobreviveu a todas as tentativas de rompimento por mais de um quarto de século e é considerado um algoritmo muito forte. Grande parte da segurança prática se baseia nele. Sua principal desvantagem é exigir chaves de pelo menos 1024 bits para manter um bom nível de segurança (em comparação com 128 bits para os algoritmos de chave simétrica), e isso o torna bastante lento.

O método RSA se baseia em alguns princípios da teoria dos números. Agora vamos mostrar de forma resumida como usar o método; para obter mais detalhes, consulte o documento original.

1. Escolha dois números primos extensos, p e q (geralmente, de 1024 bits).

2. Calcule $n = p \times q$ e $z = (p - 1) \times (q - 1)$.

3. Escolha um número *d* tal que *z* e *d* sejam primos entre si.

4. Encontre *e* de forma que $e \times d = 1 \ mod \ z$.

Com esses parâmetros calculados com antecedência, estamos prontos para começar a criptografia. Divida o texto simples (considerado um string de bits) em blocos, de modo que cada mensagem de texto simples *P* fique no intervalo $0 \leq P < n$. Isso pode ser feito agrupando-se o texto simples em blocos de *k* bits, onde *k* é o maior inteiro para o qual a desigualdade $2^k < n$ é verdadeira.

Para criptografar a mensagem *P*, calcule $C = P^e$ (mod *n*). Para descriptografar *C*, calcule $P = C^d$ (mod *n*). É possível provar que, para todo *P* na faixa especificada, as funções de criptografia e descriptografia são inversas entre si. Para realizar a criptografia, você precisa de *e* e *n*, enquanto para a descriptografia, são necessários *d* e *n*. Portanto, a chave pública consiste no par (*e*, *n*) e a chave privada consiste em (*d*, *n*).

A segurança do método se baseia na dificuldade de fatorar números extensos. Se pudesse fatorar o valor *n* (publicamente conhecido), o criptoanalista poderia então encontrar *p* e *q* e, a partir desses, encontrar *z*. Com o conhecimento de *z* e *e*, é possível encontrar *d* utilizando-se o algoritmo de Euclides. Felizmente, os matemáticos têm tentado fatorar números extensos por pelo menos 300 anos, e o conhecimento acumulado sugere que o problema é extremamente difícil.

De acordo com Rivest e seus colegas, a fatoração de um número de 500 dígitos requer 10^{25} anos, usando-se a força bruta. Nesse caso, eles pressupõem o melhor algoritmo conhecido e um computador com um tempo por instrução de 1 µs. Mesmo que os computadores continuem a se tornar cada vez mais rápidos na proporção de uma ordem de magnitude por década, ainda se passarão séculos até que a fatoração de um número de 500 dígitos se torne viável e, nesse tempo, nossos descendentes poderão simplesmente escolher *p* e *q* ainda maiores.

Um exemplo didático muito comum do algoritmo RSA é mostrado na Figura 8.17. Para esse exemplo, escolhemos $p = 3$ e $q = 11$, o que gera $n = 33$ e $z = 20$. Um valor adequado para d é $d = 7$, visto que 7 e 20 não têm fatores comuns. Com essas opções, e pode ser encontrado resolvendo-se a equação $7e = 1$ (mod 20), que produz $e = 3$. O texto cifrado C para uma mensagem de texto simples P é dado por $C = P^3$ (mod 33). O texto cifrado é decodificado pelo receptor usando a regra $P = C^7$ (mod 33). A figura mostra a codificação do texto simples "SUZANNE" como exemplo.

[arte: ver original p. 754]

[Dísticos]

[1] Texto simples (P)

Simbólico	Numérico
S	19
U	21
Z	26
A	01
N	14
N	14
E	05

[2] Texto cifrado (C)

P^3	P^3 (mod 33)
6859	28
9261	21
17576	20
1	1
2744	5
2744	5

125 26

[3]

C^Z

13492928512

1801088541

1280000000

1

78125

78125

8031810176

[4] Depois da decodificação

C^Z (mod 33) Simbólico

19 S

21 U

26 Z

1 A

14 N

14 N

5 E

[5] Cálculo do transmissor Cálculo do receptor

[F]Figura 8.17

[FL] Um exemplo do algoritmo RSA

Como os números primos escolhidos para esse exemplo são muito pequenos, *P* tem de ser menor que 33; portanto, cada bloco do texto simples só pode conter um caractere isolado. O resultado é uma cifra de substituição monoalfabética, que não impressiona muito. Se em vez disso, tivéssemos escolhido *p* e $q \approx 2^{512}$,

teríamos $n \approx 2^{1024}$ e assim cada bloco poderia ter até 1024 bits ou 128 caracteres de 8 bits, em comparação com 8 caracteres para o DES e 16 caracteres para o AES.

Devemos destacar que o uso do RSA da forma como descrevemos é semelhante ao uso de um algoritmo simétrico no modo ECB — o mesmo bloco de entrada gera o mesmo bloco de saída. Portanto, é necessário algum tipo de encadeamento para a criptografia de dados. No entanto, na prática, a maior parte dos sistemas baseados no RSA utiliza a criptografia de chave pública principalmente para distribuir chaves únicas de sessão que, por sua vez, são empregadas com algum algoritmo de chave simétrica, como AES ou DES triplo. O RSA é lento demais para codificar grandes volumes de dados, mas é amplamente utilizado para a distribuição de chaves.

[T3] 8.3.2 Outros algoritmos de chave pública

Apesar de ser amplamente utilizado, o RSA não é de forma alguma o único algoritmo de chave pública conhecido. O primeiro algoritmo de chave pública foi o algoritmo da mochila (Merkle e Hellman, 1978). A idéia aqui é que alguém possui um grande número de objetos, cada objeto com um peso diferente. O dono dos objetos codifica a mensagem selecionando secretamente um subconjunto dos objetos e colocando-os na mochila. O peso total dos objetos contidos na mochila torna-se público, e o mesmo acontece com a lista de todos os objetos possíveis. A lista de objetos contidos na mochila é mantida em segredo. Com outras restrições específicas, o problema de descobrir uma lista de objetos possíveis com o peso fornecido foi considerado computacionalmente inviável e formou a base do algoritmo de chave pública.

O inventor do algoritmo, Ralph Merkle, estava bastante seguro de que esse algoritmo não poderia ser decifrado; portanto, ofereceu um prêmio de 100

dólares a quem conseguisse fazê-lo. Adi Shamir (o "S" do RSA) se prontificou a decifrar o algoritmo e ganhou o prêmio. Indignado, Merkle reforçou o algoritmo e ofereceu um prêmio de 1.000 dólares a quem pudesse decifrá-lo. Ronald Rivest (o "R" do RSA) também conseguiu decifrar o novo algoritmo e ganhou o prêmio. Merkle não ousou oferecer 10.000 dólares pela nova versão revisada; portanto, "A" (Leonard Adleman) não teve sorte. Apesar de ter sido refeito, o algoritmo da mochila não é mais considerado seguro e não é usado.

Outros esquemas de chave pública se baseiam na dificuldade de calcular logaritmos discretos. Os algoritmos que utilizam esse princípio foram criados por El Gamal (1985) e Schnorr (1991).

Existem alguns outros esquemas, como os que se baseiam em curvas elípticas (Menezes e Vanstone, 1993), mas as duas principais categorias são aquelas que se baseiam na dificuldade de fatorar números extensos e no cálculo de logaritmos discretos cuja base é um número primo extenso. Esses problemas são considerados genuinamente difíceis de resolver — os matemáticos estão estudando os algoritmos há anos sem grandes resultados.

[T2] 8.4 Assinaturas digitais

A autenticidade de muitos documentos legais, financeiros e outros documentos é determinada pela presença de uma assinatura autorizada. Isso não vale para as fotocópias. Para que os sistemas de mensagens computadorizadas possam substituir o transporte físico de documentos em papel e tinta, deve-se encontrar um método que permita assinar os documentos de um modo que não possa ser forjado.

O problema de se criar um substituto para as assinaturas escritas à mão é muito difícil. Basicamente, necessita-se de um sistema através do qual uma parte possa enviar uma mensagem "assinada" para outra parte de forma que:

1. O receptor possa verificar a identidade alegada pelo transmissor.

2. Posteriormente, o transmissor não possa repudiar o conteúdo da mensagem.

3. O receptor não tenha a possibilidade de inventar ele mesmo a mensagem.

Por exemplo, o primeiro requisito diz respeito a sistemas financeiros. Quando o computador de um cliente pede ao computador de um banco que compre uma tonelada de ouro, o computador do banco precisa se certificar de que o computador que está emitindo o pedido realmente pertence à empresa cuja conta deve ser debitada.

O segundo requisito é necessário para proteger o banco contra fraudes. Suponha que o banco compre a tonelada de ouro e que logo depois disso o preço do ouro caia abruptamente. Um cliente desonesto poderia processar o banco, afirmando nunca ter feito qualquer pedido para a compra de ouro. Quando o banco mostra a mensagem no tribunal, o cliente nega tê-la enviado. A propriedade segundo a qual nenhuma parte de um contrato pode negar mais tarde tê-lo assinado é chamada **não repúdio.** Os esquemas de assinatura digital que estudaremos agora ajudam a garantir o não repúdio.

O terceiro requisito é necessário para proteger o cliente caso o preço do ouro dispare e o banco tente montar uma mensagem assinada na qual o cliente pedia uma barra de ouro em vez de uma tonelada. Nesse cenário de fraude, o bando simplesmente guarda para si próprio o restante do ouro.

[T3] 8.4.1 Assinaturas de chave simétrica

Uma estratégia para as assinaturas digitais é ter uma autoridade central que saiba de tudo e na qual todos confiem, digamos, Big Brother (*BB*). Em seguida, cada usuário escolhe uma chave secreta e a leva para o escritório de *BB*. Assim, somente Alice e *BB* conhecem a chave secreta de Alice, K_A e assim por diante. Quando deseja enviar uma mensagem em texto simples assinada, *P*, ao gerente

de sua conta, que é Bob, Alice gera K_A (B, R_A, t, P), onde B é a identidade de Bob, R_A é um número aleatório escolhido por Alice, t é um timbre de hora para assegurar a atualidade e K_A (B, R_A, t, P) é a mensagem criptografada com sua chave, K_A. Em seguida, ela envia a mensagem, da maneira descrita na Figura 8.18. *BB* vê que essa mensagem veio de Alice, descriptografa a mensagem e a envia a Bob, exatamente como mostramos. A mensagem para Bob contém o texto simples da mensagem de Alice e também a mensagem assinada K_{BB} (A, t, P). Agora Bob executa a solicitação de Alice.

O que acontecerá se mais tarde Alice negar ter enviado a mensagem? Na etapa 1, todo mundo processa todo mundo (pelo menos, nos Estados Unidos). Por fim, quando o caso parar nos tribunais e Alice continuar negando ter enviado a mensagem a Bob, o juiz perguntará a Bob como ele pode ter certeza de que a mensagem veio de Alice e não de Trudy. Primeiro, Bob destaca que *BB* só aceitará uma mensagem de Alice se ela tiver sido criptografada com K_A. Portanto, não há possibilidade de Trudy ter enviado a *BB* uma mensagem falsa no lugar de Alice, sem que *BB* detectasse esse farto de imediato.

[arte: ver original p. 757]

[Dísticos]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.18

[FL] Assinaturas digitais com Big Brother

Em seguida, drasticamente, Bob apresenta a Prova A, K_{BB}(A, t, P). Bob afirma tratar-se de uma mensagem assinada por *BB* que prova o fato de Alice ter enviado P a Bob. Em seguida, o juiz solicita que *BB* (em quem todos confiam)

descriptografe a Prova A. Quando *BB* testemunha que Bob está dizendo a verdade, o juiz decide a favor de Bob. Caso encerrado.

Um problema potencial com o protocolo de assinatura da Figura 8.22 é Trudy repetir uma das mensagens. Para minimizar esse problema, são utilizados timbres de hora em todas as mensagens. Além disso, Bob pode verificar todas as mensagens mais recentes para ver se R_A foi usada em alguma delas. Caso isso tenha acontecido, a mensagem será descartada por ser uma repetição. Observe que Bob rejeitará as mensagens muito antigas ao verificar seus timbres de hora. Para se proteger contra ataques de repetição repentinos, Bob verifica R_A em cada uma das mensagens recebidas para ver se a mensagem foi enviada por Alice durante a última hora. Caso contrário, Bob pode pressupor com toda segurança que essa solicitação é nova.

[T3] 8.4.2 Assinaturas de chave pública

Um problema estrutural com o uso da criptografia de chave simétrica para assinaturas digitais é que todos têm de confiar em Big Brother. Além disso, Big Brother tem de ler todas as mensagens assinadas. Os candidatos mais lógicos à execução do servidor de Big Brother são o governo, os bancos, os contadores e os advogados. Infelizmente, nenhuma dessas organizações inspira total confiança a todos os cidadãos. Daí, seria interessante se o ato de assinatura de documentos não exigisse a presença de uma autoridade confiável.

Felizmente, a criptografia de chave pública pode trazer uma importante contribuição para esse caso. Vamos supor que os algoritmos de criptografia e descriptografia de chave pública tenham a propriedade de que $E(D(P)) = P$ além, é claro, da propriedade habitual de que $D(E(P)) = P$. (O RSA tem essa propriedade; portanto, a suposição não é irracional.) Supondo-se que seja esse o caso, Alice pode enviar uma mensagem de texto simples assinada, P, para Bob transmitindo

$E_B(D_A(P))$. Observe que Alice conhece sua própria chave de descriptografia (privada), D_A, assim como a chave pública de Bob, E_B; portanto, a criação dessa mensagem é algo que Alice pode fazer.

Quando recebe a mensagem, Bob a transforma usando sua chave privada e produz $D_A(P)$, como mostra a Figura 8.19. Ele guarda esse texto em um lugar seguro e depois aplica E_A para obter o texto simples original.

[arte: ver original p. 758]

[Dísticos]

[1] Computador de Alice

P Chave privada de Alice, D_A Chave pública de Bob, E_B

$D_A(P)$

[2] Linha de transmissão

$E_B(D_A(P))$

[3] Computador de Bob

Chave privada de Bob, D_B Chave pública de Alice, E_A P

$D_A(P)$

[F]Figura 8.19

[FL] Assinaturas digitais com o uso da criptografia de chave pública

Para ver como a propriedade de assinatura funciona, suponha que posteriormente Alice negue ter enviado a mensagem *P* para Bob. Quando o caso chegar aos tribunais, Bob poderá produzir tanto *P* quanto $D_A(P)$. O juiz pode confirmar com facilidade que Bob certamente tem uma mensagem válida criptografada por D_A simplesmente aplicando E_A à mensagem. Como Bob não sabe qual é a chave privada de Alice, a única forma de Bob ter adquirido uma mensagem criptografada por essa chave seria se Alice de fato a tivesse enviado. Enquanto estiver presa por perjúrio e fraude, Alice terá bastante tempo para inventar novos

algoritmos de chave pública muito interessantes.

Apesar da utilização da criptografia de chave pública para assinaturas digitais ser um esquema elegante, existem problemas relacionados ao ambiente no qual elas operam e não ao algoritmo básico. De um lado, Bob só poderá provar que uma mensagem foi enviada por Alice enquanto D_A permanecer secreta. Se Alice revelar sua chave secreta, o argumento deixará de existir, pois qualquer um poderia ter enviado a mensagem, inclusive o próprio Bob.

Por exemplo, pode ocorrer um problema se Bob for o corretor de ações de Alice. Alice solicita a Bob que ele compre ações ou títulos de uma determinada empresa. Logo depois disso, o preço cai abruptamente. Para repudiar a mensagem que enviou a Bob, Alice vai à polícia afirmando que sua casa foi assaltada, e o PC que continha sua chave foi roubado. Dependendo das leis do estado ou do país onde mora, ela poderá ou não ser legalmente processada, em especial se afirmar que só descobriu o roubo quando chegou em casa após o trabalho, muitas horas depois do ocorrido.

Outro problema com o esquema de assinatura é o que acontecerá se Alice decidir alterar sua chave. Isso é legal, e provavelmente é uma boa idéia fazê-lo de vez em quando. Se mais tarde surgir um caso jurídico, como descrevemos antes, o juiz aplicará a E_A *atual* a $D_A(P)$ e descobrirá que ela não produz P. Nesse momento, a situação de Bob ficará complicada.

Em princípio, qualquer algoritmo de chave pública pode ser usado para assinaturas digitais. O padrão *de facto* do setor é o algoritmo RSA. Muitos produtos de segurança o utilizam. No entanto, em 1991, o NIST (National Institute of Standards and Technology) propôs a utilização de uma variante do algoritmo de chave pública de El Gamal em seu novo padrão, o **DSS (Digital Signature Standard)**. O El Gamal obtém sua segurança a partir da dificuldade de calcular logaritmos discretos, e não da dificuldade de fatorar números muito

extensos.

Como sempre acontece quando o governo tenta ditar padrões criptográficos, houve um tumulto. O DSS foi criticado por ser:

1. Secreto demais (a NSA projetou o protocolo para utilizar o El Gamal).
2. Novo demais (o El Gamal ainda não foi amplamente analisado).
3. Lento demais (10 a 40 vezes mais lento do que o RSA para verificar assinaturas).
4. Inseguro demais (chave fixa de 512 bits).

Em uma versão posterior, o quarto ponto rendeu muita discussão quando foram permitidas chaves com até 1024 bits. Apesar disso, os dois primeiros pontos permanecem válidos.

[T3] 8.4.3 Sumários de mensagens

Uma crítica aos métodos de assinatura é a de que com freqüência eles reúnem duas funções distintas: autenticação e sigilo. Em geral, a autenticação é necessária, mas o sigilo não. Como a criptografia é lenta, normalmente as pessoas preferem enviar documentos em texto simples assinados. A seguir, descreveremos um esquema de autenticação que não exige a criptografia da mensagem inteira.

Esse esquema se baseia na idéia de uma função de hash unidirecional que extrai um trecho qualquer do texto simples e a partir dele calcula um string de bits de tamanho fixo. Essa função de hash, representada por *MD*, geralmente é chamada **sumário de mensagens** e tem quatro importantes propriedades:

1. Se *P* for fornecido, o cálculo de $MD(P)$ será muito fácil.
2. Se $MD(P)$ for fornecido, será efetivamente impossível encontrar *P*.
3. Dado *P*, ninguém pode encontrar *P'* tal que $MD(P') = MD(P)$.
4. Uma mudança na entrada de até mesmo 1 bit produz uma saída muito

diferente.

Para atender ao critério 3, a função de hash deve ter pelo menos 128 bits, de preferência mais. Para atender ao critério 4, o hash deve @@@desfigurar completamente os bits, o que não é diferente dos algoritmos de criptografia de chave simétrica que vimos.

Calcular um sumário de mensagens a partir de um trecho de texto simples é muito mais rápido que criptografar esse texto simples com um algoritmo de chave pública; portanto, os sumários de mensagens podem ser usados para agilizar algoritmos de assinatura digital. Para ver como isso funciona, considere mais uma vez o protocolo de assinatura da Figura 8.18. Em vez de assinar P com $K_{BB}(A, t, P)$, agora BB calcula a compilação da mensagem aplicando MD a P, produzindo $MD(P)$. Em seguida, BB inclui $K_{BB}(A, t, MD(P))$ como o quinto item da lista criptografada com K_B que é enviada a Bob, em vez de $K_{BB}(A, t, P)$.

Se houver uma contestação, Bob poderá produzir tanto P quanto $K_{BB}(A, t, MD(P))$. Depois que Big Brother tiver descriptografado o item para o juiz, Bob terá $MD(P)$ que, certamente, é genuíno, além do P alegado. No entanto, como é impossível para Bob encontrar outra mensagem que produza esse hash, o juiz será facilmente convencido de que Bob está falando a verdade. A utilização de sumários de mensagens dessa forma poupa tempo de criptografia e reduz os custos com o armazenamento e o transporte de mensagens.

Além disso, sumários de mensagens funcionam muito bem em sistemas de criptografia de chave pública, como mostra a Figura 8.20. Aqui, primeiro Alice calcula o sumário de mensagens de seu texto simples. Em seguida, ela assina a mensagem e envia tanto a compilação assinada quando o texto simples a Bob. Se Trudy substituir P durante a operação, Bob perceberá a troca quando calcular $MD(P)$.

[arte: ver original p. 760]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.20

[FL] Assinaturas digitais utilizando sumários de mensagens

[T4] MD5

Foram propostas diversas funções de sumário de mensagens. As mais amplamente utilizadas são o MD5 (Rivest, 1992) e SHA-1 (NIST, 1993). O **MD5** é o quinto de uma série de sumários de mensagens criadas por Ronald Rivest. Ele opera desfigurando os bits de uma forma tão complicada que todos os bits de saída são afetados por todos os bits de entrada. Resumindo, a função começa aumentando o tamanho da mensagem até chegar a 448 bits (módulo 512). Em seguida, o tamanho original é anexado como um inteiro de 64 bits, a fim de gerar uma entrada total cujo tamanho seja um múltiplo de 512 bits. A última etapa antes dos cálculos serem efetuados é inicializar um buffer de 128 bits com um valor fixo.

Agora começam os cálculos. Em cada rodada, um bloco de entrada de 512 bits é extraído e colocado no buffer de 128 bits. Para que os cálculos sejam feitos com maior precisão, também é incluída uma tabela criada a partir da função de seno. O objetivo da utilização de uma função conhecida, como o seno, é evitar qualquer suspeita de que o projetista tenha criado uma armadilha secreta para seu próprio uso, e não pelo fato dessa função ser mais aleatória do que um gerador de números aleatórios. A recusa da IBM em revelar os princípios em que se baseava o projeto das caixas S do DES criou muita especulação sobre esses artifícios secretos. Há quatro rodadas para cada bloco de entrada. Esse processo continua até que todos os blocos de entrada tenham sido consumidos. O conteúdo do

buffer de 128 bits forma o sumário de mensagens.

O MD5 já é usado há mais de uma década, e muitos pessoas o têm atacado. Foram encontradas algumas vulnerabilidades, mas certas etapas internas impedem que ele seja rompido. Porém, se as barreiras restantes dentro do MD5 caírem, ele poderá falhar eventualmente. Apesar disso, no momento em que escrevemos, ele ainda resiste.

[T4] SHA-1

A outra função importante do sumário de mensagens é o **SHA-1 (Secure Hash Algorithm 1)**, desenvolvido pela NSA e aprovado pelo NIST no FIPS 180-1. A exemplo do MD5, esse algoritmo processa os dados de entrada em blocos de 512 bits. No entanto, ao contrário do MD5, ele gera um sumário de mensagens de 160 bits. Um modo típico de Alice enviar uma mensagem não secreta mas assinada para Bob é ilustrado na Figura 8.21. Nessa figura, sua mensagem de texto simples é colocada no algoritmo SHA-1 para se obter um hash de 160 bits do SHA-1. Em seguida, Alice assina o hash com sua chave privada RSA e envia a mensagem de texto simples e o hash assinado para Bob.

[arte: ver original p. 761]

[Dísticos]

[1] Mensagem M de Alice em texto simples (tamanho arbitrário)

[2] Algoritmo SHA-1

[3] Hash SHA-1 de 160 bits de M

H

[4] Chave privada de Alice, D_A

Algoritmo RSA

[5] Hash assinado

$D_A(H)$

[6] Enviado para Bob

[F]Figura 8.21

[FL] Utilização do SHA-1 e do RSA para assinar mensagens não secretas

Depois de receber a mensagem, o próprio Bob calcula o hash SHA-1 e também aplica a chave pública de Alice ao hash assinado para obter o hash original, *H*. Se os dois corresponderem, a mensagem será considerada válida. Tendo em vista que não há nenhum meio para Trudy modificar a menagem (de texto simples) enquanto ela estiver em trânsito e produzir uma nova mensagem com hash *H*, Bob pode detectar com facilidade quaisquer mudanças que Trudy tenha feito na mensagem. Para mensagens cuja integridade é importante, mas cujo conteúdo não é secreto, o esquema da Figura 8.21 é bastante utilizado. Por um custo relativamente pequeno em computação, ele garante que as modificações feitas na mensagem de texto simples em trânsito poderão ser detectadas com probabilidade muito alta.

Agora, vamos ver rapidamente como o SHA-1 funciona. Ele começa preenchendo a mensagem com a adição de um bit 1 ao final, seguido pelo número de bits 0 necessários para tornar o tamanho um múltiplo de 512 bits. Em seguida, um número de 64 bits contendo o tamanho da mensagem antes do preenchimento é submetido a uma operação OR nos 64 bits de baixa ordem. Na Figura 8.22, a mensagem é mostrada com o preenchimento à direita, porque o texto em inglês e as figuras se estendem da esquerda para a direita (isto é, o canto inferior direito em geral é percebido como o fim da figura). Com os computadores, essa orientação corresponde a máquinas big-endian, como SPARC, mas o SHA-1 sempre preenche o fim da mensagem, não importando que máquina endian seja usada.

Durante a computação, o SHA-1 mantém cinco variáveis de 32 bits, de H_0 a H_4,

onde o hash se acumula. Essas variáveis são mostradas na Figura 8.22(b). Elas são inicializadas como constantes especificadas no padrão.

Cada um dos blocos M_0 a M_{n-1} é agora processado. Para o bloco atual, primeiro as 16 palavras são copiadas no início de um array auxiliar de 80 palavras, *W*, como mostra a Figura 8.22(c). Em seguida, as outras 64 palavras em *W* são preenchidas, usando-se a fórmula:

[Inserir equação do O.A. p. 762a]

onde $S^b(W)$ representa a rotação circular à esquerda da palavra de 32 bits, *W*, por *b* bits. Agora, são inicializados cinco variáveis de rascunho de *A* até *E*, com valores de H_0 a H_4, respectivamente.

[arte: ver original p. 762]

[Dísticos]

[1] Início da mensagem Bloco de 512 bits Palavra de 32 bits

[2] M_0 H_0 W_0

M_1 H_1 W_1

M_2 H_2 W_2

Preenchimento H_3

M_{n-1} H_4 W_{79}

(a) (b) (c)

[F]Figura 8.22

[FL] (a) Uma mensagem preenchida até um múltiplo de 512 bits. (b) As variáveis de saída. (c) O array de palavras

O cálculo real pode ser expresso em pseudo-C como:

[TD]

```
for (i = 0; i <80; i++) {
    temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
```

E = D; D = C; C = S^{30}(B); B = A; A = temp;

} [TN]

onde as K_i constantes são definidas no padrão. As funções misturadas f_i são definidos como:

[Inserir equação do O.A. p. 762b]

Quando todas as 80 iterações do loop são completadas, as varáveis de *A* até *E* são somadas a H_0 até H_4, respectivamente.

Agora que o primeiro bloco de 512 bits foi processado, o próximo é iniciado. O array *W* é reinicializado a partir do novo bloco, mas *H* fica como estava. Quando esse bloco é concluído, o próximo é iniciado e assim por diante, até todos os blocos da mensagem de 512 bits serem processados. Quando o último bloco é concluído, as cinco palavras de 32 bits no array *H* são transmitidas como saída, formando o hash criptográfico de 160 bits. O código C completo para SHA-1 é dado na RFC 3174.

Novas versões de SHA-1 estão em desenvolvimento para hashes de 256, 384 e 512 bits, respectivamente.

[T3] 8.4.4 Ataque de aniversário

No mundo da criptografia, nada é o que parece. Talvez você esteja pensando que são necessárias aproximadamente 2^m operações para subverter um sumário de mensagens de *m* bits. Na verdade, normalmente $2^{m/2}$ operações serão suficientes, utilizando-se o **ataque de aniversário**, um método publicado por Yuval (1979) no clássico artigo "How to Swindle Rabin" (Como enganar Rabin).

A idéia para esse ataque vem de uma técnica que freqüentemente os professores de matemática utilizam em seus cursos de probabilidade. A pergunta é a seguinte: quantos alunos você deverá ter em uma sala de aula para que a probabilidade de haver duas pessoas fazendo aniversário no mesmo dia exceda

1/2? A maioria dos alunos espera que a resposta seja superior a 100. Na verdade, a teoria da probabilidade afirma que esse número é apenas 23. Intuitivamente, sem fazer uma análise rigorosa, com 23 pessoas podemos formar $(23 \times 22)/2 = 253$ pares diferentes, e cada um dos quais tem uma probabilidade igual a 1/365 de ser um acerto. Sob esse aspecto, essa probabilidade deixa de ser tão surpreendente.

Mais genericamente, se houver algum mapeamento entre as entradas e as saídas, com n entradas (pessoas, mensagens etc.) e k saídas possíveis (aniversários, sumários de mensagens etc.), haverá $n(n - 1)/2$ pares de entrada. Se $n(n - 1)/2 > k$, a chance de haver pelo menos uma correspondência será muito boa. Portanto, fazendo uma aproximação, é provável que haja uma correspondência para [ver símbolo]. Esse resultado significa que provavelmente um sumário de mensagens de 64 bits possa ser rompido gerando-se 2^{32} mensagens e procurando-se duas mensagens com o mesmo sumário.

Agora vejamos um exemplo prático. O departamento de ciência da computação da State University nos Estados Unidos tem uma única vaga estável para dois professores, Tom e Dick. Tom foi admitido dois anos antes de Dick e, portanto, é convocado para os testes primeiro. Se ele for aprovado, Dick será descartado. Tom sabe que a chefe de seu departamento, Marilyn, gosta muito do trabalho dele; sendo assim, ele pede que Marilyn escreva uma carta de recomendação ao reitor, que tomará a decisão a respeito do cargo. Depois de enviadas, todas as cartas passam a ser confidenciais.

Marilyn pede que sua secretária, Ellen, escreva a carta para o reitor, fazendo um esboço do que deseja. Quando a carta está pronta, Marilyn a revisa, calcula e assina o sumário de 64 bits e o envia ao reitor. Ellen pode enviar a carta mais tarde através de correio eletrônico.

Infelizmente para Tom, Ellen está envolvida emocionalmente com Dick e gostaria

que Tom fosse descartado; assim, escreve a carta a seguir com as 32 opções entre colchetes.

Caro Sr. Reitor,

Esta [*carta* | *mensagem*] tem como objetivo expressar minha [*honesta* | *franca*] opinião a respeito do professor Tom Wilson, que [*é* | *está*] [*candidato* | *prestes*] [*a* | *para*] obter uma vaga permanente nesta universidade [*imediatamente* | *este ano*]. Eu [*conheço* | *trabalho com*] o professor Wilson há seis anos. Ele é um [*destacado* | *excelente*] pesquisador de grande [*talento* | *capacidade*] [*mundialmente* | *internacionalmente*] conhecido por suas [*brilhantes* | *criativas*] idéias a respeito de [*muitos* | *uma grande variedade de*] problemas [*difíceis* | *complicados*].

Ele também é um [*professor* | *educador*] [*bastante* | *muito*] [*respeitado* | *admirado*]. Seus alunos fazem críticas [*maravilhosas* | *espetaculares*] de suas [*aulas* | *cursos*]. Ele é o [*professor* | *orientador*] [*mais querido* | *mais conhecido*] [*da universidade* | *do departamento*].

[*Além disso* | *Além do mais*], o professor Wilson é um [*grande* | *fantástico*] administrador. [*Seus* | *Suas*] [*contratos* | *concessões*] trouxeram uma [*grande* | *substancial*] quantia em dinheiro para [*o* | *nosso*] departamento. [*Esse dinheiro* | *Esses fundos*] [*permitiu* | *permitiram*] que [*criássemos* | *realizássemos*] muitos programas [*especiais* | *importantes*], [*tais como* | *por exemplo*] o programa Universidade 2000. Sem esses fundos, [*seríamos incapazes* | *não seríamos capazes*] de dar continuidade a esse programa, que é tão [*importante* | *essencial*] para nós. Afirmo ao senhor que ele é o profissional mais adequado para essa posição.

Infelizmente para Tom, assim que acaba de redigir e digitar essa carta, Ellen também digita a seguinte carta:

Caro Sr. Reitor,

Esta [*carta* | *mensagem*] tem como objetivo expressar minha [*honesta* | *franca*] opinião a respeito do professor Tom Wilson, que [*é* |*está*] [*candidato* | *prestes*] [*a* / *para*] assumir uma vaga permanente nesta universidade [*imediatamente* | *este ano*]. Eu [*conheço* | *trabalho com*] Tom há seis anos. Ele é um [*incompetente* | *mau*] pesquisador, não é bem visto em sua [*especialidade* | *área*]. Sua pesquisa [*raramente* |*esporadicamente*] mostra [*bom-senso* | *conhecimento*] dos [*principais* | *mais importantes*] problemas atuais.

Ele não é um [*professor* | *educador*] [*bastante* | *muito*] [*respeitado* |*admirado*]. Seus alunos fazem [*duras* | *pesadas*] críticas de suas [*aulas* | *cursos*]. Ele é o [*professor* | *orientador*] mais impopular [*da universidade* | *do departamento*] devido a sua [*tendência* | *propensão*] a [*ridicularizar* | *embaraçar*] os alunos que fazem perguntas em suas aulas.

[*Além disso* | *Além do mais*], Tom é um administrador [*terrível* | *fraco*]. [*Seus* | *Suas*] [*contratos* | *concessões*] trouxeram apenas uma [*insignificante* | *pequena*] quantia em dinheiro para [*o* | *nosso*] departamento. A menos que mais [*verbas* | *fundos*] sejam [*alocadas* | *alocados*] , teremos de cancelar alguns programas essenciais, tais como o seu programa Universidade 2000. Infelizmente, sob essas [*condições* | *circunstâncias*], não posso recomendá-lo em sã consciência para essa posição.

Ellen passa a noite configurando seu computador para calcular os 2^{32} sumários de mensagens de cada carta. Há chances de que um sumário da primeira carta corresponda a um sumário da segunda carta. Caso isso não aconteça, ela poderá incluir algumas outras opções e tentar de novo durante o fim de semana. Suponha que ela encontre uma correspondência. Vamos chamar a carta "boa" de *A* e a "ruim" de *B*.

Em seguida, através do correio eletrônico, Ellen envia a carta *A* a Marilyn para que seja aprovada. Ela mantém a carta *B* completamente secreta, sem mostrá-la a

ninguém. É claro que Marilyn aprova a carta, calcula seu sumário de mensagens de 64 bits, assina o sumário e envia o sumário assinado ao reitor Smith utilizando o correio eletrônico. Por outro lado, Ellen envia a carta *B* ao reitor (não a carta *A*, como deveria fazer).

Depois de obter a carta e o sumário de mensagens assinado, o reitor executa o algoritmo de sumário de mensagens na carta *B*, observa que ela corresponde ao sumário que Marilyn enviou e despede Tom. O reitor não percebe que Ellen gerou duas cartas com o mesmo sumário de mensagens e enviou a ele uma mensagem diferente da que Marylin viu e aprovou. (Final opcional: Ellen conta a Dick o que fez. Dick não gosta do que ouve e termina o namoro com ela. Ellen fica furiosa e confessa tudo a Marilyn. Marilyn telefona para o reitor. Tom acaba ficando com o cargo.) Com o MD5, o ataque de aniversário se torna impraticável, pois mesmo com um bilhão de sumários por segundo, seriam necessários 500 anos para calcular 2^{64} sumários de duas cartas com 64 variantes cada um e, de qualquer forma, não poderíamos ter certeza de que haveria uma correspondência. É claro que, com 5000 computadores trabalhando em paralelo, 500 anos se transformam em cinco semanas. O SHA-1 é melhor (por ser mais longo).

[T2] 8.5 Gerenciamento de chaves públicas

A criptografia de chave pública torna possível a comunicação segura para pessoas que não compartilham uma chave comum, e também possibilita a assinatura de mensagens sem a presença de uma terceira parte confiável. Finalmente, os sumários de mensagens assinados permitem verificar com facilidade a integridade de mensagens recebidas.

Porém, existe um problema que ignoramos até aqui: se Alice e Bob não conhecem um ao outro, como ele irão obter as respectivas chaves públicas para iniciar o processo de comunicação? A solução óbvia — colocar a chave pública no Web site

— não funciona pela seguinte razão: suponha que Alice queira pesquisar a chave pública de Bob em seu Web site. Como ela fará isso? Bem, Alice começa digitando o URL de Bob. Seu navegador então pesquisa o endereço DNS da home page de Bob e envia a ele uma solicitação *GET*, como mostra a Figura 8.23. Infelizmente, Trudy intercepta a solicitação e responde com uma home page falsa, talvez uma cópia da home page de Bob, exceto pela substituição da chave pública de Bob pela chave pública de Trudy. Quando Alice codifica sua primeira mensagem com E_T, Trudy a decodificará, lerá e recodificará com a chave pública de Bob, enviando a mensagem a Bob, que não sabe que Trudy está lendo suas mensagens recebidas. Pior ainda, Trudy poderia modificar as mensagens antes de recodificá-las para Bob. É claro que há necessidade de algum mecanismo para garantir que as chaves públicas possam ser trocadas em segurança.

[arte: ver original p. 765]

[Dísticos]

[1] Alice

[2]

1. GET home page de Bob

2. Falsifica home page com E_T

3. E_T(Mensagem)

[3] Trudy

[4] 4. E_B(Mensagem)

[5] Bob

[F]Figura 8.23

[FL] Um modo de Trudy subverter a criptografia de chave pública

[T3] 8.5.1 Certificados

Como uma primeira tentativa de distribuição de chaves públicas com segurança,

poderíamos imaginar um centro de distribuição de chaves disponível on-line 24 horas por dia a fim de fornecer chaves públicas por demanda. Um dos muitos problemas com essa solução é o fato de ela não ser escalável, e o centro de distribuição de chaves rapidamente se tornaria um gargalo. Além disso, se ele ficasse inativo, a segurança da Internet seria paralisada repentinamente.

Por essas razões, as pessoas desenvolveram uma solução diferente, que não exige que o centro de distribuição de chaves esteja on-line todo o tempo. De fato, ele não precisa estar on-line de modo algum. Em vez disso, ele certifica as chaves públicas pertencentes a pessoas, empresas e outras organizações. Uma organização que certifica chaves públicas é chamada **CA (Certification Authority — autoridade de certificação)**.

Como um exemplo, suponha que Bob queira permitir que Alice e outras pessoas se comuniquem com ele em segurança. Ele pode ir até a CA com sua chave pública e seu passaporte ou com a carteira de motorista e solicitar a certificação. A CA emite então um certificado semelhante ao da Figura 8.24 e assina seu hash SHA-1 com a chave privada da CA. Em seguida, Bob paga a taxa da CA e obtém um disquete contendo o certificado e seu hash assinado.

[arte: ver original p. 766]

I hereby certify that the public key

1 9836A8B030300F83737E3837837FC3587O92827262643FFA8271

0382828282A

belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4,1958

Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA's private key

[F]Figura 8.24

[FL] Um certificado possível e seu hash assinado

A principal função de um certificado é vincular uma chave pública ao nome de um protagonista (indivíduo, empresa etc.). Os certificados em si não são secretos ou protegidos. Por exemplo, Bob poderia decidir colocar seu novo certificado em seu Web site, com um link na página principal informando: clique aqui para ver meu certificado de chave pública. O clique resultante retornaria o certificado e o bloco de assinatura (o hash SHA-1 assinado do certificado).

Agora vamos percorrer o cenário da Figura 8.23 novamente. Quando Trudy intercepta a solicitação de Alice para a home page de Bob, o que ela pode fazer? Trudy pode inserir seu próprio certificado e seu bloco de assinatura na página falsa; porém, quando Alice ler o certificado, verá imediatamente que não está se comunicando com Bob, porque o nome de Bob não está no certificado. Trudy pode modificar a home page de Bob durante a execução, substituindo a chave pública de Bob pela sua própria chave. Porém, quando Alice executar o algoritmo SHA-1 no certificado, ela obterá um hash que não concorda com o que ela recebe ao aplicar a chave pública conhecida da CA ao bloco de assinatura. Como Trudy não tem a chave privada da CA, ela não tem meios de gerar um bloco de assinatura que contenha o hash da página da Web modificada com sua chave pública. Desse modo, Alice pode estar certa de que possui a chave pública de Bob e não a de Trudy ou de outra pessoa. Como prometemos, esse esquema não exige que a CA esteja on-line para verificação, eliminando assim um gargalo potencial.

Embora a função padrão de um certificado seja vincular uma chave pública a um protagonista, um certificado também pode ser usado para vincular uma chave

pública a um **atributo**. Por exemplo, um certificado poderia afirmar: esta chave pública pertence a alguém com mais de 18 anos. Ela pode ser usada para provar que o proprietário da chave privada não é menor de idade e, portanto, pode acessar material não apropriado para crianças e assim por diante, mas sem revelar a identidade do proprietário. Em geral, a pessoa que tivesse o certificado o enviaria ao Web site, ao protagonista ou ao processo que solicitasse informações sobre a idade. Esse site, protagonista ou processo, geraria então um número aleatório e o codificaria com a chave pública no certificado. Se o proprietário fosse capaz de decodificá-lo e enviá-lo de volta, essa seria aprova de que o proprietário de fato tinha o atributo declarado no certificado. Como alternativa, o número aleatório poderia ser usado para gerar uma chave de sessão pela duração da conversação.

Outro exemplo de situação em que um certificado poderia conter um atributo é um sistema distribuído orientado a objetos. Em geral, cada objeto tem diversos métodos. O proprietário do objeto poderia fornecer a cada cliente um certificado com um mapa de bits dos métodos que o cliente tem permissão para invocar e vincular o mapa de bits a uma chave pública, usando um certificado assinado. Mais uma vez, se o proprietário do certificado puder provar a posse da chave privada correspondente, ele terá permissão para executar os métodos no mapa de bits. Ele não precisa conhecer a identidade do proprietário, uma característica útil em situações nas quais a privacidade é importante.

[T3] 8.5.2 X.509

Se todo mundo quisesse que algo assinado foi enviado a CA com um tipo de certificado diferente, logo se tornaria um problema administrar todos os formatos diferentes. Para resolver esse problema, foi criado e aprovado pela ITU um padrão para certificados. O padrão é chamado **X.509** e seu uso está difundido na

Internet. Ele passou por três versões desde a padronização inicial em 1988.

Vamos descrever a V3.

O X.509 foi fortemente influenciado pelo mundo OSI, tomando emprestadas algumas de suas piores características (por exemplo, nomenclatura e codificação). Surpreendentemente a IETF aceitou o X.509, embora em quase todas as outras áreas — desde endereços de máquina até protocolos de transporte e formatos de correio eletrônico — ela tenha ignorado a OSI e tentado fazer tudo da maneira certa. A versão da IETF do X.509 é descrita na RFC 3280.

Em seu núcleo, o X.509 é um modo de descrever certificados. Os principais campos em um certificado estão listados na Figura 8.25. As descrições dadas na figura devem fornecer uma idéia geral do significado dos campos. Para obter informações adicionais, consulte o próprio padrão ou a RFC 2459.

Por exemplo, se Bob trabalhar no departamento de empréstimos do Money Bank, seu endereço X.500 poderá ser:

[TD] /C=US/O=MoneyBank/OU=Loan/CN=Bob/ [TN]

onde *C* é o país, *O* é a organização, *OU* é a unidade organizacional e *CN* é o nome comum. As CAs e outras entidades são identificadas de modo semelhante. Um problema significativo com os nomes X.500 é que, se Alice estiver tentando entrar em contato com *bob@moneybank.com* e receber um certificado com um nome X.500, talvez não fique claro para ela a que Bob o certificado se refere. Felizmente, a partir da versão 3, os nomes DNS são permitidos, em lugar de nomes X.500; assim, esse problema eventualmente deve desaparecer.

[arte: ver original p. 768]

[T]Tabela

Campo

Version

Serial number

Signature algorithm

Issuer

Validity period

Subject name

Public key

Issuer ID

Subject ID

Extensions

Signature

Significado

A versão do X.509

Este número, somado ao nome da CA, identifica de forma exclusiva o certificado

O algoritmo usado para assinar o certificado

Nome X.500 da CA

A hora inicial e final do período de validade

A entidade cuja chave está estando certificada

A chave pública do assunto e a ID do algoritmo que a utiliza

Uma ID opcional que identifica de forma exclusiva o emissor do certificado

Uma ID opcional que identifica de forma exclusiva o assunto do certificado

Muitas extensões foram definidas

A assinatura do certificado (assinado pela chave privada da CA)

[F]Figura 8.25

[FL] Os campos básicos de um certificado X.509

Os certificados são codificados com o uso da **ASN.1 (Abstract Syntax Notation 1)** da OSI, que pode ser considerada uma struct em C, exceto por sua notação muito peculiar e extensa. Para obter mais informações sobre o X.509, consulte (Ford e

Baum, 2000).

[T3] 8.5.3 Infra-estruturas de chave pública

Fazer uma única CA emitir todos os certificados do mundo evidentemente não funcionaria. Ela entraria em colapso sob a carga e também seria um ponto central de falha. Uma solução possível poderia ser a existência de várias CAs, todas administradas pela mesma organização e todas usando a mesma chave privada para assinar certificados. Embora isso pudesse resolver o problema da carga e da falha, há um novo problema: o vazamento de chaves. Se houvesse dezenas de servidores espalhados pelo mundo, todos com a chave privada da CA, a chance de que a chave privada fosse roubada ou sofresse algum outro tipo de vazamento seria bastante aumentada. Tendo em vista que o comprometimento dessa chave arruinaria a infra-estrutura de segurança eletrônica do mundo, a existência de uma única CA central é muito arriscada.

Além disso, que organização operaria a CA? É difícil imaginar uma autoridade que fosse aceita em todo o mundo como uma entidade legítima e confiável. Em alguns países, as pessoas insistiriam em que essa entidade fosse um governo; em outros países, elas insistiriam que não fosse um governo.

Por essas razões, foi desenvolvido um modo diferente de certificar chaves públicas, identificada pelo nome geral **PKI (Public Key Infrastructure).** Nesta seção, resumiremos como ela funciona em linhas gerais, embora existam muitas propostas relativas aos detalhes que provavelmente irão evoluir com o tempo.

Uma PKI tem vários componentes, incluindo usuários, CAs, certificados e diretórios. A função da PKI é fornecer um modo de estruturar esses componentes e definir padrões para os vários documentos e protocolos. Uma forma particularmente simples de PKI é uma hierarquia de CAs, como mostra a Figura 8.26. Nesse exemplo, mostramos três níveis mas, na prática, pode haver um

número menor ou maior. A CA de nível superior, chamada raiz, certifica CAs do segundo nível, que denominamos **RAs (Regional Authorities)**, porque podem cobrir alguma região geográfica, como um país ou um continente. Entretanto, esse termo não é padrão; de fato, nenhum termo é realmente padrão para os diferentes níveis da árvore. Por sua vez, as RAs certificam as CAs reais, que emitem os certificados X.509 para organizações e indivíduos. Quando a raiz autoriza uma nova RA, ela gera um certificado X.509 anunciando que aprovou a RA, inclui a chave pública da nova RA no certificado, assina o certificado e o entrega à RA. De modo semelhante, quando uma RA aprova uma nova CA, ela produz e assina um certificado declarando sua aprovação e contendo a chave pública da CA.

Nossa PKI funciona de modo semelhante. Suponha que Alice precise da chave pública de Bob, a fim de se comunicar com ele; então, ela procura e encontra um certificado contendo a chave, assinado pela CA 5. Porém, Alice nunca ouviu falar da CA 5. Tudo que ela sabe é que a CA 5 pode ser a filha de 10 anos de Bob. Ela poderia ir até a CA 5 e dizer: prove sua legitimidade. A CA 5 responde com o certificado que recebeu da RA 2, que contém a chave pública da CA 5. Agora, munida da chave pública da CA 5, Alice pode confirmar que o certificado de Bob foi de fato assinado pela CA 5 e, portanto, é válido.

A menos que a RA 2 seja o filho de 12 de Bob. Nesse caso, a próxima etapa é pedir a RA 2 que prove sua legitimidade. A resposta à consulta de Alice é um certificado assinado pela raiz e contendo a chave pública da RA 2. Agora, Alice tem certeza de que possui a chave pública de Bob.

[arte: ver original p. 769]

[Dísticos]

[1] Raiz

[2] A RA 2 é aprovada. Sua chave pública é 47383AE349...

Assinatura da raiz

[3] A CA 5 é aprovada. Sua chave pública é 6384AF863B...

Assinatura da RA 2

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.26

[FL] (a) Uma PKI hierárquico. (b) Uma cadeia de certificados

No entanto, como Alice encontra a chave pública da raiz? Por mágica. Supõem-se que todo mundo conhece a chave pública da raiz. Por exemplo, seu navegador pode ter sido comercializado com a chave pública da raiz embutida.

Bob é o tipo de sujeito amigável e não quer dar muito trabalho a Alice. Ele sabe que Alice vai ter de verificar a CA 5 e a RA 2; assim, para evitar dificuldades, ele reúne os dois certificados necessários e os fornece a ela juntamente com o seu próprio certificado. Agora, ela pode usar seu conhecimento da chave pública da raiz para confirmar o certificado de nível superior e a chave pública que ele contém para verificar o segundo certificado. Desse modo, Alice não precisa entrar em contato com ninguém para fazer a verificação. Como os certificados são todos assinados, ela pode detectar com facilidade quaisquer tentativas de falsificar seu conteúdo. Uma cadeia de certificados como essa que volta à raiz, às vezes é chamada **cadeia de confiança** ou **caminho de certificação.** A técnica é amplamente utilizada na prática.

É claro que ainda temos o problema de saber quem vai administrar a raiz. A solução é não ter uma única raiz, mas sim várias raízes, cada uma com suas próprias RAs e CAs. De fato, os navegadores modernos são pré-carregados com as chaves públicas de mais de 100 raízes, às vezes referidas como **âncoras de**

confiança. Desse modo, pode-se evitar ter uma única autoridade confiável no mundo inteiro.

Entretanto, agora existe a questão de como o fornecedor do navegador decide quais das supostas âncoras de confiança são de fato confiáveis e quais são desprezíveis. Tudo se reduz à confiança do usuário no fornecedor do navegador para fazer escolhas sensatas e não aprovar simplesmente todas as âncoras de confiança dispostas a pagar por sua inclusão na lista. A maioria dos navegadores permite que os usuários inspecionem as chaves da raiz (em geral, sob a forma de certificados assinados pela raiz) e eliminem qualquer uma que parecer obscura.

[T4] Diretórios

Outra questão importante para qualquer PKI é onde estão armazenados os certificados (e suas cadeias de retorno até alguma âncora de confiança conhecida). Uma possibilidade é fazer cada usuário armazenar seus próprios certificados. Embora isso seja seguro (isto é, não existe nenhum meio dos usuários adulterarem certificados assinados sem detecção), também é inconveniente. Uma alternativa proposta é usar o DNS como um diretório de certificados. Antes de entrar em contato com Bob, é provável que Alice tenha de pesquisar seu endereço IP usando o DNS; então, por que não fazer o DNS retornar toda a cadeia de certificados de Bob juntamente com seu endereço IP?

Algumas pessoas acham que essa é a melhor alternativa, mas outras talvez prefiram servidores de diretórios dedicados cuja única tarefa seja administrar certificados X.509. Tais diretórios poderiam fornecer serviços de pesquisa usando propriedades dos nomes X.500. Por exemplo, na teoria tal serviço de diretório poderia transferir uma consulta como: "Forneça uma lista de todas as pessoas chamadas Alice que trabalham em departamentos de vendas de algum lugar nos Estados Unidos ou no Canadá". O LDAP poderia conter tais informações.

[T4] Revogação

O mundo real também está repleto de certificados, como de passaportes e carteiras de motoristas. Às vezes, esses certificados podem ser revogados, bem como as carteiras de motoristas que são flagrados dirigindo bêbedos ou cometendo outras infrações de trânsito. O mesmo problema ocorre no mundo digital: a autoridade que concede um certificado pode decidir revogá-lo porque a pessoa ou organização que possui o certificado cometeu algum abuso. Ele também pode ser revogado se a chave privada foi exposta ou, pior ainda, se a chave privada da CA foi comprometida. Desse modo, uma PKI precisa lidar com a questão da revogação.

Um primeiro passo nessa direção é fazer cada CA emitir periodicamente uma **CRL (Certificate Revocation List — lista de revogação de certificados)** fornecendo os números de série de todos os certificados que ela revogou. Tendo em vista que os certificados contêm prazos de validade, a CRL só precisa conter os números de série de certificados ainda não vencidos. Uma vez que seu prazo de validade tenha passado, um certificado se torna automaticamente inválido, e assim não há necessidade de distinção entre os que alcançaram o prazo limite e os que foram de fato revogados. Em ambos os casos, eles não podem mais ser utilizados.

Infelizmente, a introdução de CRLs significa que um usuário prestes a usar um certificado deve agora adquirir a CRL para ver se o certificado foi revogado. Se foi, ele não deve ser usado. Porém, mesmo que o certificado não esteja na lista, ele poderia ter sido revogado logo após a lista ter sido publicada. Desse modo, a única forma de realmente ter certeza é consultar a CA. Além disso, no próximo uso do certificado, a CA tem de ser consultada de novo, pois o certificado poderia ter sido revogado alguns segundos antes.

Outra complicação é o fato de um certificado revogado poder ser reabilitado, por

exemplo, se tiver sido revogado por não pagamento de alguma taxa que foi paga posteriormente. Ter de lidar com a revogação (e talvez com a reabilitação) elimina uma das melhores propriedades dos certificados, ou seja, a possibilidade de usá-los sem ter de entrar em contato com uma CA.

Onde as CRLs devem ser armazenadas? Um boa alternativa seria armazená-las no mesmo local em que estão os próprios certificados. Uma estratégia é a CA publicar ativamente CRLs periódicas e fazer os diretórios processá-las apenas removendo os certificados revogados. Se os diretórios não forem usados para armazenar os certificados, as CRLs poderão ser guardadas no cache em diversos locais convenientes na rede. Tendo em vista que uma CRL também é um documento assinado, se ela for adulterada, essa ação poderá ser facilmente detectada.

Se os certificados tiverem uma longa duração, as CRLs também serão longas. Por exemplo, se os cartões de crédito forem válidos por 5 anos, o número de revogações pendentes será muito maior do que seria se fossem emitidos novos cartões a cada 3 meses. Um modo padrão de lidar com CRLs longas é emitir uma lista mestre com pouca freqüência, mas emitir atualizações freqüentes para a lista. Isso reduz a largura de banda necessária para distribuir as CRLs.

[T2] 8.6 Segurança da comunicação

Agora, concluímos nosso estudo das principais ferramentas. A maior parte das técnicas e protocolos importantes foi abordada. O restante do capítulo estuda a aplicação dessas técnicas na prática para proporcionar segurança às redes, além de alguns conceitos sobre os aspectos sociais da segurança, no final do capítulo.

Nas quatro seções a seguir, examinaremos a segurança da comunicação, isto é, como levar os bits secretamente e sem alteração da origem até o destino, e como manter bits indesejáveis do lado de fora. Essas não são de modo algum as únicas

questões de segurança em redes, mas certamente estão entre as mais importantes, o que nos dá um bom ponto de partida.

[T3] 8.6.1 IPsec

A IETF sabia há muitos anos que havia carência de segurança na Internet. Não era fácil aumentá-la, porque havia uma disputa para definir onde colocá-la. A maioria dos especialistas em segurança acredita que, para serem realmente seguras, a criptografia e as verificações de integridade devem ser realizadas de fim a fim (isto é, na camada de aplicação). Desse modo, o processo de origem criptografa e/ou protege a integridade dos dados e os envia ao processo de destino, onde eles são descriptografados e/ou verificados. Qualquer adulteração realizada entre esses dois processos, inclusive dentro de qualquer sistema operacional, poderá então ser detectada. A dificuldade com essa abordagem é que ela exige a troca de todas as aplicações, a fim de torná-las cientes da segurança. Nessa visão, a segunda melhor abordagem é inserir a criptografia na camada de transporte ou em uma nova camada entre a camada de aplicação e a camada de transporte, tornando-a ainda fim a fim, mas sem exigir que as aplicações sejam alteradas.

A visão oposta é a de que os usuários não entendem de segurança e não serão capazes de usá-la corretamente e, como ninguém quer modificar programas existentes, a camada de rede devia autenticar e/ou codificar pacotes sem os usuários estejam envolvidos. Depois de anos de batalhas, essa visão ganhou apoio suficiente para que fosse definido um padrão de segurança da camada de rede. Em parte, o argumento era de que realizar a codificação na camada de rede não impediria que usuários conscientes da segurança a implementassem na camada de aplicação e, até certo ponto, isso também poderia ajudar aos usuários sem consciência da segurança.

O resultado dessa guerra foi um projeto chamado **IPsec (IP security)**, descrito nas RFCs 2401, 2402 e 2406, entre outras. Nem todos os usuários desejam a criptografia (porque ela é dispendiosa em termos computacionais).Em vez de torná-la opcional, decidiu-se exigir a criptografia o tempo todo, mas permitir o uso de um algoritmo nulo. O algoritmo nulo é descrito e louvado por sua simplicidade, facilidade de implementação e grande velocidade na RFC 2410.

O projeto completo do IPsec é uma estrutura para vários serviços, algoritmos e granularidades. A razão para vários serviços é que nem todo mundo quer pagar o preço de ter todos os serviços o tempo todo, e assim os serviços estão disponíveis à escolha de cada usuário. Os principais serviços são sigilo, integridade de dados e proteção contra ataques de reprodução (o intruso reproduz uma conversação). Todos esses serviços se baseiam na criptografia de chave simétrica, porque o alto desempenho é importante.

Existem vários algoritmos porque um algoritmo que agora é considerado seguro poderá ser violado no futuro. Tornando o IPsec independente do algoritmo, a estrutura pode sobreviver até mesmo se algum algoritmo específico for violado mais tarde.

A razão para várias granularidades é tornar possível a proteção de uma única conexão TCP, de todo tráfego entre um par de hosts ou de todo o tráfego entre um par de roteadores seguros, além de outras possibilidades.

Um aspecto um tanto surpreendente do IPsec é que, embora esteja na camada IP, ele é orientado a conexões. Na realidade, isso não é muito surpreendente porque, para ter qualquer segurança, uma chave tem de ser estabelecida e usada por algum período de tempo — em essência, um espécie de conexão. Além disso, as conexões amortizam os custos de configuração por vários pacotes. Uma "conexão" no contexto do IPsec é chamada **SA (security association)**. Uma SA é uma conexão simplex entre dois pontos extremos e tem um identificador de

segurança associado a ela. Se houver necessidade de tráfego seguro em ambos os sentidos, será exigidas duas associações de segurança. Os identificadores de segurança são transportados em pacotes e percorrem essas conexões seguras e são usados para pesquisar chaves e outras informações relevantes ao chegar um pacote seguro.

Tecnicamente, o IPsec tem duas partes principais. A primeira parte descreve dois novos cabeçalhos que podem ser acrescentados a pacotes, a fim de transportar o identificador de segurança, os dados de controle de integridade e outras informações. A outra parte, o **ISAKMP (Internet Security Association and Key Management Protocol)** lida com o estabelecimento de chaves. Não trataremos mais do ISAKMP porque (1) ele é extremamente complexo e (2) seu principal protocolo, o **IKE (Internet Key Exchange)**, tem falhas profundas e precisa ser substituído (Perlman e Kaufman, 2000).

O IPsec pode ser usado em dois modos. No **modo de transporte**, o cabeçalho IPsec é inserido logo depois do cabeçalho IP. O campo *Protocol* no cabeçalho IP é alterado para indicar que um cabeçalho IPsec segue o cabeçalho IP normal (antes do cabeçalho TCP). O cabeçalho IPsec contém informações de segurança, principalmente o identificador SA, um novo número de seqüência e, possivelmente, uma verificação de integridade da carga útil.

No **modo de túnel**, todo o pacote IP, incluindo o cabeçalho, é encapsulado no corpo de um novo pacote IP com um cabeçalho IP completamente novo. O modo de túnel é útil quando o túnel termina em um local diferente do destino final. Em alguns casos, o fim do túnel é uma máquina de gateway de segurança, por exemplo, um firewall da empresa. Nesse modo, o firewall encapsula e desencapsula pacotes à medida que eles passam pelo firewall. Quando o túnel termina nessa máquina segura, as máquinas da LAN da empresa não têm de tomar conhecimento do IPsec. Isso é tarefa do firewall.

O modo de túnel também é útil quando um conjunto de conexões TCP é agregado e tratado como um único fluxo codificado, porque isso evita que um intruso veja quem está enviando, quem está recebendo e quantos pacotes são enviados. Às vezes, o simples conhecimento da quantidade de tráfego e de seu destino é uma informação valiosa. Por exemplo, se durante uma crise militar o volume de tráfego que flui entre o Pentágono e a Casa Branca cair de forma abrupta, mas o volume de tráfego entre o Pentágono e alguma instalação militar nas profundezas das Montanhas Rochosas do Colorado aumentar na mesma proporção, um intruso poderá deduzir algumas informações úteis desses dados. O estudo dos padrões de fluxo de pacotes, ainda que eles estejam codificados, é chamado **análise de tráfego.** O modo de túnel fornece um meio para anular até certo ponto essa análise. A desvantagem do modo de túnel é que ele acrescenta um cabeçalho IP extra, aumentando assim substancialmente o tamanho dos pacotes. Em contraste, o modo de transporte não afeta muito o tamanho dos pacotes.

O primeiro cabeçalho novo é **AH (Authentication Header).** Ele fornece verificação de integridade e segurança contra reprodução, mas não oferece sigilo (isto é, não há criptografia de dados). O uso do AH no modo de transporte é ilustrado na Figura 8.27. No IPv4, ele é inserido entre o cabeçalho IP (incluindo quaisquer opções) e o cabeçalho TCP. No lPv6 ele é simplesmente outro cabeçalho de extensão e é tratado como tal. De fato, o formato é próximo ao de um cabeçalho de extensão padrão do IPv6. É possível que a carga útil tenha de ser preenchida até completar algum tamanho específico para o algoritmo de autenticação, como mostra a figura.

[arte: ver original p. 774]

[Dísticos]

[1] Autenticado

[2] Cabeçalho lP AH Cabeçalho TCP Carga útil + preenchimento

[3]32 Bits

[4] Next header Payload len (Reservado)

[5] Security parameters index

[6] Sequence number

[7] Authentication data (HMAC)

[F]Figura 8.27

[FL] O cabeçalho de autenticação do IPsec em modo de transporte para o IPv4

Agora, vamos examinar o cabeçalho AH. O campo *Next header* é usado para armazenar o valor anterior que o campo *Protocol* do IP tinha antes de ser substituído por 51 para indicar que haverá um cabeçalho AH em seguida. Na maioria dos casos, o código para o TCP (6) ficará aqui. O campo *Payload length* é o número de palavras de 32 bits no cabeçalho AH, menos 2 unidades.

O campo *Security parameters index* é o identificador da conexão. Ele é inserido pelo transmissor para indicar um registro específico no banco de dados do receptor. Esse registro contém a chave compartilhada usada nessa conexão e outras informações sobre a conexão. Se esse protocolo tivesse sido criado pela ITU e não pela IETF, esse campo seria chamado *Virtual circuit number*.

O campo *Sequence number* é usado para numerar todos os pacotes enviados em uma SA. Todo pacote recebe um número exclusivo, mesmo as retransmissões. Em outras palavras, a retransmissão de um pacote recebe um número diferente do original (embora seu número de seqüência do TCP seja o mesmo). O propósito desse campo é detectar ataques de reprodução. Esses números de seqüência não podem se repetir. Se todos os 2^{32} se esgotarem, terá de ser estabelecida uma nova SA para dar continuidade à comunicação.

Finalmente, chegamos ao campo *Authentication data*, um campo de comprimento variável, que contém a assinatura digital da carga útil. Quando a SA é

estabelecida, os dois lados negociam o algoritmo de assinatura que irão usar. Normalmente, não é utilizada aqui a criptografia de chave pública, porque os pacotes devem ser processados de forma extremamente rápida e todos os algoritmos de chave pública conhecidos são lentos demais. Como o IPsec se baseia na criptografia de chave simétrica, e como o transmissor e o receptor negociam uma chave compartilhada antes de instalar uma SA, a chave compartilhada é usada no cálculo da assinatura. Um modo simples é calcular o hash sobre o pacote, somado à chave compartilhada. É claro que a chave compartilhada não é transmitida. Um esquema como esse é chamado **HMAC (Hashed Message Authentication Code)**. É muito mais rápido calcular o valor desse esquema que executar primeiro o SHA-1 e depois executar o RSA sobre o resultado.

O cabeçalho AH não permite criptografia dos dados; portanto, ele é útil principalmente quando a verificação da integridade é necessária, mas não o sigilo. Uma característica do AH que vale a pena notar é que a verificação de integridade abrange alguns dos campos do cabeçalho IP, ou seja, aqueles que não se alteram à medida que o pacote passa de um roteador para roteador. Por exemplo, o campo *Time to live* muda a cada hop e assim não pode ser incluído na verificação de integridade. Contudo, o endereço de origem IP é incluído na verificação, o que torna impossível para um intruso falsificar a origem de um pacote.

O cabeçalho IPsec alternativo é a **ESP (Encapsulating Security Payload)**. Seu uso no modo de transporte e no modo de túnel é mostrado na Figura 8.28.

[arte: ver original p. 775]

[Dísticos]

[1] Autenticado

[2] (a) Cabeçalho IP Cabeçalho ESP Cabeçalho TCP Carga útil +

Preenchimento Autenticação (HMAC)

Criptografado

[3] Autenticado

[4] (b) Novo cabeçalho IP Cabeçalho ESP Antigo cabeçalho IP Cabeçalho TCP Carga útil + Preenchimento Autenticação (HMAC)

Criptografado

[F]Figura 8.28

[FL] (a) ESP em modo de transporte. (b) ESP em modo de túnel

O cabeçalho ESP consiste em duas palavras de 32 bits. Elas constituem os campos *Security parameters index* e *Sequence number* que vimos no AH. Uma terceira palavra que geralmente segue esses campos (mas tecnicamente não faz parte do cabeçalho) é o *Initialization vector* usado para criptografia de dados, a menos que seja utilizada a criptografia nula pois, nesse caso, ele será omitido.

A ESP também fornece verificações de integridade do HMAC, como o AH; porém, em vez de serem incluídas no cabeçalho, elas vêm depois da carga útil, como mostra a Figura 8.28. A colocação do HMAC no final tem uma vantagem em uma implementação de hardware. O HMAC pode ser calculado à medida que os bits saem pela interface de rede e são acrescentados ao final. Por essa razão, as redes Ethernet e outras LANs têm seus CRCs em um final (trailer), e não em um cabeçalho. Com o AH, o pacote tem de ser armazenado no buffer e a assinatura deve ser calculada antes de ser possível enviar o pacote, reduzindo potencialmente o número de pacotes/s que podem ser enviados.

Considerando que a ESP pode fazer tudo que o AH pode fazer e muito mais, além de ser mais eficiente durante a inicialização, surge a questão: afinal, qual é a necessidade do AH? A resposta é principalmente histórica. No início, o AH cuidava apenas da integridade, enquanto a ESP tratava do sigilo. Mais tarde, a integridade

foi acrescentada à ESP, mas as pessoas que projetaram o AH não queriam deixá-lo morrer depois de tanto trabalho. No entanto, o único argumento real dessas pessoas se baseava no fato de que o AH é capaz de verificar uma parte do cabeçalho IP, o que a ESP não faz. Porém, esse é um argumento fraco, como também o argumento de que um produto com suporte para o AH, mas não para a ESP, poderia ter menos problemas para obter uma licença de exportação, porque não poderia efetuar a codificação. É provável que o AH fique defasado no futuro.

[T3] 8.6.2 Firewalls

A capacidade de conectar qualquer computador em qualquer lugar a qualquer outro computador em qualquer lugar é uma faca de dois gumes. É muito divertido para as pessoas navegarem pela Internet quando estão em casa. Para os gerentes de segurança das empresas, trata-se de um pesadelo. Muitas empresas têm grandes quantidades de informações confidenciais on-line —segredos comerciais, planos de desenvolvimento de produtos, estratégias de marketing, análises financeiras etc. A revelação dessas informações para um concorrente poderia ter terríveis conseqüências.

Além do perigo das informações virem a público, também há o perigo do vazamento dessas informações dentro da empresa. Em particular, vírus, vermes e outras pestes digitais podem burlar a segurança, destruir dados valiosos e consumir muito tempo dos administradores, que tentam eliminar a confusão causada por eles. Com freqüência, eles são trazidos por funcionários descuidados que querem brincar com algum jogo novo muito divertido.

Em conseqüência disso, são necessários mecanismos para manter os "bons" bits e descartar os "maus" bits. Um dos métodos é usar o IPsec, que protege os dados em trânsito entre sites seguros. No entanto, o IPsec não faz nada para impedir as pestes digitais e os intrusos de invadirem a LAN da empresa. Para ver como

alcançar esse objetivo, precisamos examinar os firewalls.

[arte: ver original p. 777]

[Dísticos]

[1] Roteador de filtragem de pacotes

[2] Gateway de aplicação

[3] Roteador de filtragem de pacotes

[4] Conexões para redes externas

[5] Firewall

[6] LAN externa

[7] LAN interna

[8] Perímetro de segurança

[9] Rede corporativa

[10] Backbone

[F]Figura 8.29

[FL] Um firewall que consiste em dois filtros de pacotes e um gateway de aplicação

Os **firewalls** são apenas uma adaptação moderna de uma antiga forma de segurança medieval: cavar um fosso profundo em torno do castelo. Esse recurso forçava todos aqueles que quisessem entrar ou sair do castelo a passar por uma única ponte levadiça, onde poderiam ser revistados por guardas. Nas redes, é possível usar o mesmo artifício: uma empresa pode ter muitas LANs conectadas de forma arbitrária, mas todo o tráfego de saída ou de entrada da empresa é feito através de uma ponte levadiça eletrônica (firewall), como mostra a Figura 8.29.
O firewall, nessa configuração, tem dois componentes: dois roteadores que filtram pacotes e um gateway de aplicação. Também existem configurações mais simples, mas a vantagem desse projeto é que cada pacote deve passar por dois

filtros e um gateway de aplicação para sair ou entrar na rede. Não existem outras rotas. Os leitores que consideram um único ponto de conferência suficiente para fins de segurança, sem dúvida não têm viajado em vôos internacionais nos últimos meses.

Cada **filtro de pacotes** é um roteador padrão equipado com algumas funções complementares, que permitem a inspeção de cada pacote de entrada ou de saída. Os pacotes que atenderem a algum critério serão remetidos normalmente, mas os que falharem no teste serão descartados.

Na Figura 8.29, provavelmente o filtro de pacotes da LAN interna verificará os pacotes enviados, e o da LAN externa conferirá os pacotes recebidos. Os pacotes que atravessam o primeiro obstáculo vão para o gateway de aplicação, onde serão submetidos a uma nova verificação. A colocação dos dois filtros de pacotes em diferentes LANs destina-se a assegurar que nenhum pacote entre ou saia sem passar pelo gateway de aplicação, pois não há outro caminho.

Em geral, os filtros de pacotes são baseados em tabelas configuradas pelo administrador do sistema. Essas tabelas listam as origens e os destinos aceitáveis, as origens ou destinos bloqueados e as regras padrão que orientam o que deve ser feito com os pacotes recebidos de outras máquinas ou destinados a elas.

No caso comum de uma configuração TCP/IP, uma origem ou destino consiste em uma porta e um endereço IP. As portas indicam qual é o serviço desejado. Por exemplo, a porta 23 do TCP é para telnet, a porta 79 é para finger e a porta 119 é para notícias da USENET. Uma empresa poderia bloquear os pacotes recebidos em relação a todos os endereços IP combinados com uma dessas portas. Dessa forma, ninguém fora da empresa poderia estabelecer login via telnet ou procurar alguém usando o daemon Finger. Além disso, a empresa também evitaria que seus funcionários passassem todo o dia lendo notícias da USENET.

O bloqueio de pacotes de saída é mais complicado porque, embora muitos sites adotem as convenções padrão para numeração de portas, eles não são obrigados a fazê-lo. Além disso, para alguns serviços importantes, como FTP (File Transfer Protocol), os números de portas são atribuídos dinamicamente. Além disso, embora o bloqueio das conexões TCP seja difícil, o bloqueio de pacotes UDP é ainda mais complicado, porque se sabe muito pouco (*a priori*) sobre o que eles farão. Muitos filtros de pacotes simplesmente não aceitam tráfego UDP.

A segunda metade do mecanismo de firewall é o **gateway de aplicação.** Em vez de apenas examinar pacotes brutos, o gateway opera na camada de aplicação. Por exemplo, um gateway de correio eletrônico pode ser configurado de forma a examinar cada mensagem recebida ou enviada. O gateway toma a decisão de transmitir ou descartar cada mensagem, com base nos campos de cabeçalho, no tamanho da mensagem ou até mesmo em seu conteúdo (por exemplo, em uma instalação militar, a presença de palavras como "nuclear" ou "bomba" pode provocar algum tipo de ação especial).

As instalações têm liberdade para instalar um ou mais gateways de aplicação destinados a aplicações específicas, mas não é incomum que organizações suspeitas permitam a entrada e a saída de mensagens de correio eletrônico e até usem a World Wide Web, mas proíbam tudo que seja muito arriscado. Combinada com a criptografia e a filtragem de pacotes, essa estratégia oferece um nível de segurança limitado ao custo de algumas inconveniências.

Ainda que o firewall esteja perfeitamente configurado, ainda existem vários problemas de segurança. Por exemplo, se um firewall estiver configurado para permitir apenas a entrada de pacotes de redes específicas (por exemplo, outras fábricas da empresa), um intruso fora do firewall pode inserir falsos endereços de origem para ultrapassar essa verificação. Se um usuário interno quiser transportar documentos secretos para fora da empresa, ele poderá codificar ou

até mesmo fotografar os documentos e transportar as fotografias como arquivos JPEG, que conseguirão passar por quaisquer filtros de palavras. Não discutimos nem mesmo o fato de que 70% de todos os ataques vêm de dentro do firewall; por exemplo, de funcionários insatisfeitos (Schneier, 2000).

Além disso, há toda uma classe de diferentes ataques com que os firewalls não podem lidar. A idéia básica por trás de um firewall é impedir a entrada de intrusos e a saída de dados secretos. Infelizmente, existem pessoas que não têm nada melhor para fazer do que tentar derrubar certos sites. Para isso, eles enviam ao destino pacotes legítimos em grande quantidade, até o site entrar em colapso com a carga. Por exemplo, para incapacitar um Web site, um intruso pode enviar um pacote *SYN* do TCP para estabelecer uma conexão. Então, o site alocará um slot de tabela para a conexão e enviará um pacote *SYN* + *ACK* em resposta. Se o intruso não responder, o slot de tabela ficará retido por alguns segundos até o timeout. Se o intruso enviar milhares de solicitações de conexão, todas os slots de tabela serão preenchidos e nenhuma conexão legítima poderá passar. Os ataques em que o objetivo do intruso é desativar o destino em vez de roubar dados são chamados ataques **DoS (Denial of Service — negação de serviço)**. Em geral, os pacotes solicitados têm endereços de origem falsos, para que o intruso não possa ser rastreado com facilidade.

Uma variante ainda pior é aquela em que o intruso já entrou em centenas de computadores em outros lugares do mundo, e depois comanda todos esses computadores em um ataque ao mesmo alvo ao mesmo tempo. Essa estratégia não apenas aumenta o poder de fogo do intruso, mas também reduz a chance de detecção, pois os pacotes estão vindo de um grande número de máquinas pertencentes a usuários insuspeitos. Um ataque desse tipo é chamado **DDoS (Distributed Denial of Service)**, e é muito difícil proteger-se contra ele. Ainda que a máquina atacada pode reconhecer rapidamente uma solicitação falsa, processar

e descartar a solicitação é um processo que leva algum tempo e, se chegarem solicitações em número suficiente por segundo, a CPU passará todo seu tempo lidando com elas..

[T3] 8.6.3 Redes privadas virtuais

Muitas empresas têm escritórios e fábricas espalhados por muitas cidades, às vezes por vários países. Antigamente, antes das redes públicas de dados, era comum tais empresas arrendarem linhas dedicadas da companhia telefônica entre alguns pares de locais ou entre todos eles. Algumas empresas ainda fazem isso. Uma rede construída a partir de computadores de empresas e de linhas telefônicas dedicadas é chamada **rede privada.** Um exemplo de rede privada que conecta três locais é mostrado na Figura 8.30(a).

As redes privadas funcionam muito bem e são bastante seguras. Se as únicas linhas disponíveis forem as linhas dedicadas, nenhum tráfego poderá vazar para fora das instalações da empresa, e os intrusos terão de grampear fisicamente as linhas para entrar, o que não é fácil. O problema das redes privadas é que arrendar uma única linha T1 custa milhares de dólares por mês, e as linhas T3 tem um custo muito elevado. Quando surgiram as redes públicas de dados e mais tarde a Internet, muitas empresas optaram por mover seu tráfego de dados (e possivelmente o de voz) para a rede pública, mas sem desistirem da segurança da rede privada.

Essa demanda logo levou à criação de **VPNs (Virtual Private Networks)**, que são redes sobrepostas às redes públicas, mas com a maioria das propriedades de redes privadas. Elas são chamadas "virtuais" porque são meramente uma ilusão, da mesma forma que os circuitos virtuais não são circuitos reais e que a memória virtual não é memória real.

Embora as VPNs possam ser implementadas sobre redes ATM (ou frame relay),

uma abordagem cada vez mais popular é construir VPNs diretamente sobre a Internet. Um projeto comum é equipar cada escritório com um firewall e criar túneis pela Internet entre todos os pares de escritórios, como ilustra a Figura 8.30(b). Se o IPsec for usado no tunneling, será possível agregar todo o tráfego entre dois pares de escritórios quaisquer em uma única SA autenticada e criptografada, fornecendo assim controle de integridade, sigilo e até mesmo uma considerável imunidade à análise de tráfego.

[arte: ver original p. 779]

[Dísticos]

[1] Escritório 1 Linha dedicada Escritório 2

Escritório 3

(a)

[2] Escritório 1 Firewall Internet Escritório 2

Túnel

Escritório 3

(b)

[F]Figura 8.30

[FL] (a) Uma rede privada de linha dedicada. (b) Uma rede privada virtual

Quando o sistema é criado, cada par de firewalls tem de negociar os parâmetros de sua SA, incluindo os serviços, os modos, os algoritmos e as chaves. Muitos firewalls têm recursos internos de VPN, embora alguns roteadores comuns possam fazer isso muito bem. Porém, como os firewalls se destinam principalmente a questões de segurança, é natural fazer os túneis começarem e terminarem nos firewalls, proporcionando uma separação clara entre a empresa e a Internet. Desse modo, firewalls, VPNs e IPsec com ESP em modo de túnel formam uma combinação natural e amplamente usada na prática.

Depois que as SAs são estabelecidas, o tráfego pode começar a fluir. Para um roteador na Internet, um pacote que viaja por um túnel VPN é apenas um pacote comum. O único detalhe pouco usual sobre ele é a presença do cabeçalho IPsec depois do cabeçalho IP; porém, como esses cabeçalhos extras não têm nenhum efeito sobre o processo de encaminhamento, os roteadores não se preocupam com esse cabeçalho extra.

Uma vantagem importante dessa forma de organizar uma VPN é sua completa transparência para todo o software do usuário. Os firewalls configuram e gerenciam as SAs. A única pessoa consciente dessa configuração é o administrador de sistema, que tem de configurar e administrar os firewalls. Para todas as outras pessoas, é como ter de novo uma rede privada de linha dedicada. Para obter mais informações sobre VPNs, consulte (Brown, 1999; e Izzo, 2000).

[T3] 8.6.4 Segurança sem fios

É surpreendentemente fácil projetar um sistema com total segurança em termos lógicos usando VPNs e firewalls, muito embora na prática ele vaze como uma peneira. Essa situação pode ocorrer se algumas das máquinas forem sem fios e usarem comunicação de rádio, que passa pelo firewall em ambos os sentidos. O alcance das redes 802.11 freqüentemente é de algumas centenas de metros; assim, qualquer pessoa que queira espionar uma empresa pode dirigir até o estacionamento dos funcionários pela manhã, deixar um notebook capaz de reconhecer sinais 802.11 dentro do carro para registrar tudo que ouvir e partir no final do dia. À tarde, o disco rígido estará repleto de valiosas informações. Teoricamente, esse vazamento não deveria acontecer. Na teoria, as pessoas não deveriam roubar bancos.

Grande parte do problema de segurança pode ter sua origem nos fabricantes de estações base sem fios (pontos de acesso) que tentam tornar seus produtos

amigáveis para o usuário. Em geral, se o usuário retirar o dispositivo da caixa e o conectar à tomada da rede elétrica, ele começará a operar de imediato — quase sempre sem qualquer segurança, revelando segredos para todo mundo que estiver dentro do alcance de rádio. Se ele for conectado a uma rede Ethernet, todo tráfego da Ethernet também aparecerá de repente no estacionamento. A rede sem fio é um sonho que se tornou realidade para o espião: dados gratuitos sem qualquer trabalho. Por essa razão, não é preciso dizer que a segurança é ainda mais importante para sistemas sem fios que para sistemas fisicamente conectados. Nesta seção, examinaremos alguns aspectos de segurança das redes sem fios. Algumas informações adicionais podem ser encontradas em (Nichols e Lekkas, 2002).

[T4] Segurança de redes 802.11

O padrão 802.11 prescreve um protocolo de segurança do nível de enlace de dados, chamado **WEP (Wired Equivalent Privacy)**, projetado para tornar a segurança de uma LAN sem fio tão boa quanto a de uma LAN fisicamente conectada. Tendo em vista que o padrão para LANs fisicamente conectadas é nenhuma segurança, é fácil alcançar esse objetivo, e a WEP o alcança, como veremos.

Quando a segurança do 802.11 é ativada, cada estação tem uma chave secreta compartilhada com a estação base. A forma como as chaves são distribuídas não é especificada pelo padrão. Elas poderiam ser pré-carregadas pelo fabricante, trocadas com antecedência pela rede fisicamente conectada. Finalmente, a estação base ou máquina do usuário poderia escolher uma chave ao acaso e enviá-la à outra máquina pelo ar, codificada com a chave pública da outra máquina. Uma vez estabelecidas, em geral as chaves permanecem estáveis por meses ou anos.

A criptografia da WEP utiliza uma cifra de fluxo baseada no algoritmo RC4. O RC4 foi projetado por Ronald Rivest e se manteve secreto até vazar e ser publicado na Internet em 1994. Como assinalamos antes, é quase impossível manter os algoritmos secretos, mesmo quando o objetivo é proteger a propriedade intelectual (como nesse caso), em vez da segurança pela obscuridade (que não era o objetivo no caso do RC4). Na WEP, o RC4 gera um fluxo de chaves que sofre uma operação XOR com um texto simples para formar o texto cifrado.

Cada carga útil de pacote é codificada com a utilização do método apresentado da Figura 8.31. Primeiro, a carga útil é verificada usando-se o polinômio CRC-32, e o total de verificação é anexado à carga útil para formar o texto simples que será usado no algoritmo de criptografia. Em seguida, esse texto simples sofre uma operação XOR com um bloco de fluxo de chaves de igual tamanho. O resultado é o texto cifrado. O IV usado para iniciar o RC4 é enviado com o texto cifrado. Quando o receptor obtém o pacote, ele extrai a carga útil criptografada, gera o fluxo de chaves a partir da chave secreta compartilhada e o IV que acabou de receber, e depois efetua uma operação XOR do fluxo de chaves com a carga útil para recuperar o texto simples. Em seguida, ele pode conferir o total de verificação para ver se o pacote foi adulterado.

Embora essa abordagem pareça boa à primeira vista, um método para rompê-la já foi publicado (Borisov *et al.*, 2001). Resumiremos a seguir seus resultados. Em primeiro lugar, um número surpreendentemente grande de instalações utiliza a mesma chave compartilhada para todos os usuários e, nesse caso, cada usuário pode ler todo o tráfego dos outros usuários. Sem dúvida, isso equivale à Ethernet, mas não é muito seguro.

Porém, mesmo que cada usuário tenha uma chave distinta, a WEP ainda pode ser atacada. Como as chaves geralmente são estáveis por longos períodos, o padrão WEP recomenda (mas não obriga) que o IV seja alterado em cada pacote para

evitar o ataque de reutilização de fluxo de chaves que discutimos na Seção 8.2.3. Infelizmente, muitos cartões 802.11 para notebooks redefinem o IV como 0 quando o cartão é inserido no computador e incrementam o valor em uma unidade a cada pacote enviado. Como as pessoas com freqüência removem e reinserem esses cartões, pacotes com baixos valores de IV são comuns. Se Trudy puder coletar vários pacotes enviados pelo mesmo usuário com o mesmo valor de IV (que é enviado em texto simples juntamente com cada pacote), ela poderá calcular o XOR de dois valores de texto simples e provavelmente violar a cifra.

[arte: ver original p. 782]

[Dísticos]

[1] Texto simples

[2] Carga útil de pacote   Total de verificação

[3] XOR

[4] Fluxo de chaves gerado pelo RC4 (chave, IV)

[5] IV   Texto cifrado

[6] Dados transmitidos

[F]Figura 8.31

[FL] Codificação de pacotes usando a WEP

No entanto, ainda que o cartão 802.11 escolha um IV ao acaso para cada pacote, os IVs só têm 24 bits; portanto, depois que forem enviados 2^{24} pacotes, eles terão de ser reutilizados. Pior ainda, com os IVs escolhidos ao acaso, o número esperado de pacotes que têm de ser enviados antes de um mesmo IV ser usado duas vezes é cerca de 5000, devido ao ataque de aniversário descrito na Seção 8.4.4. Desse modo, se escutar durante alguns minutos, Trudy quase certamente conseguirá captar dois pacotes com o mesmo IV e a mesma chave. Efetuando o XOR dos textos cifrados, ela poderá obter o XOR dos textos simples. Essa

seqüência de bits pode ser atacada de várias maneiras para recuperar os textos simples. Com um pouco mais trabalho, o fluxo de chaves para esse IV também poderá ser obtido. Trudy será capaz de continuar a trabalhar assim por algum tempo e de compilar um dicionário de fluxos de chaves para diversos IVs. Depois que um IV for rompido, todos os pacotes enviados com ele no futuro (mas também no passado) poderão ser totalmente decodificados.

Além disso, como os IVs são usados ao acaso, depois que Trudy tiver determinado um par (IV, fluxo de chaves) válido, ela poderá usar esse par para gerar todos os pacotes que quiser e, desse modo, interferir ativamente com a comunicação. Na teoria, um receptor poderia notar que grandes números de pacotes surgiram de repente, todos com o mesmo IV, mas (1) a WEP permite isso e (2) ninguém se preocupa em verificar esse detalhe.

Finalmente, o CRC não compensa muito, pois é possível para Trudy alterar a carga útil e fazer a mudança correspondente no CRC, sem sequer remover a codificação. Em suma, romper a segurança do 802.11 é bastante simples, e ainda nem listamos todos os ataques que Borisov *et al.* encontraram.

Em agosto de 2001, um mês depois de Borisov *et al.* apresentarem seu artigo, foi publicado outro ataque devastador sobre a WEP (Fluhrer *et al.*, 2001). Esse ataque encontrou deficiências criptográficas no próprio RC4. Fluhrer *et al.* descobriram que muitas chaves têm a propriedade de possibilitarem a derivação de alguns bits da chave a partir do fluxo de chaves. Se esse ataque for executado repetidamente, será possível derivar a chave inteira com pouco esforço. Sendo um tanto inclinados aos aspectos teóricos, Fluhrer *et al.* na realidade não tentaram invadir nenhuma LAN 802.11.

Em contraste, quando um estudante do curso de verão e dois pesquisadores da AT&T Labs tomaram conhecimento do ataque descrito por Fluhrer *et al.*, eles decidiram experimentá-lo na prática (Stubblefield *et al.*, 2002). Em uma semana

eles romperam sua primeira chave de 128 bits em uma LAN 802.11 de produção, e a maior parte da semana foi dedicada à busca do cartão 802.11 de preço mais baixo que puderam encontrar, obter permissão para comprá-lo, e depois instalar e testar o cartão. A programação demorou apenas duas horas.

Quando eles anunciaram seus resultados, a CNN divulgou uma reportagem intitulada "Ataque com cartão comercial rompe criptografia sem fio", e alguns gurus da indústria tentaram minimizar seus resultados dizendo que o que eles tinham feito era trivial. considerando-se os resultados de Fluhrer *et al*. Embora esse comentários seja tecnicamente verdadeiro, permanece o fato de que os esforços combinados dessas duas equipes demonstraram uma falha fatal na WEP e no 802.11.

Em 7 de setembro de 2001, o IEEE respondeu ao fato da WEP ter sido completamente violada emitindo uma breve declaração de seis pontos que podem ser resumidos assim:

1. Nós avisamos que a segurança da WEP não era melhor que a da Ethernet.
2. Uma ameaça muito maior é esquecer de ativar a segurança.
3. Tente usar algum outro tipo de sistema de segurança (por exemplo, a segurança da camada de transporte).
4. A próxima versão, 802.11i, terá melhores recursos de segurança.
5. No futuro, a certificação obrigará a utilização do 802.11i.
6. Tentaremos descobrir o que fazer até a chegada do 802.11i.

Descrevemos essa história com detalhes para enfatizar que não é fácil implementar recursos de segurança, até mesmo para especialistas.

[T4] Segurança do Bluetooth

O Bluetooth tem um alcance bem mais curto que o 802.11, e portanto não pode ser atacado a partir do estacionamento, embora a segurança ainda seja uma

questão importante nesse caso. Por exemplo, imagine que o computador de Alice esteja equipado com um teclado Bluetooth sem fio. Na ausência de segurança, se Trudy estiver no escritório ao lado, ela poderá ler tudo Alice digitou, inclusive toda sua correspondência enviada. Ela também poderá captar tudo que o computador de Alice enviar à impressora Bluetooth mais próxima (por exemplo, as mensagens de correio eletrônico recebidas e os relatórios confidenciais). Felizmente, o Bluetooth tem um esquema de segurança elaborado para tentar anular as Trudies desse mundo. Agora vamos resumir as principais características desse esquema.

O Bluetooth tem três modos de segurança, variando desde nenhuma segurança até total criptografia de dados e controle de integridade. Como ocorre com o 802.11, se a segurança for desativada (o padrão), não haverá nenhuma segurança. A maioria dos usuários mantém a segurança desativada até ocorrer uma séria violação; depois eles a ativam. No mundo agrícola, essa abordagem é conhecida como trancar a porteira depois que o cavalo escapou.

O Bluetooth fornece segurança em várias camadas. Na camada física, os saltos de freqüência oferecem um nível mínimo de segurança mas, como qualquer dispositivo Bluetooth que se desloca em uma piconet tem de ser informado da seqüência de saltos de freqüência, é óbvio que essa freqüência não é um segredo. A segurança real começa quando o escravo recém-chegado solicita um canal ao mestre. Supõem-se que os dois dispositivos compartilham uma chave secreta configurada com antecedência. Em alguns casos, ambos são fisicamente conectados pelo fabricante (por exemplo, um fone de ouvido e um telefone celular vendidos como uma unidade). Em outros casos, um dispositivo (por exemplo, o fone de ouvido) tem uma chave embutida no código e o usuário tem de digitar essa chave no outro dispositivo (por exemplo, o telefone celular) como um número decimal. Essas chaves compartilhadas são chamadas @@@chaves de

passagem.

Para estabelecer um canal, o escravo e o mestre verificam se a outra máquina conhece a chave de passagem. Nesse caso, eles negociam para ver se esse canal será criptografado, terá sua integridade controlada ou ambos. Em seguida, eles selecionam uma chave de sessão aleatória de 128 bits, na qual alguns bits podem ser públicos. A razão para permitir o enfraquecimento dessa chave é obedecer a algumas restrições do governo de vários países, criadas para impedir a exportação ou o uso de chaves mais longas do que o governo pode violar.

A criptografia utiliza uma cifra de fluxo chamada E_0; o controle de integridade emprega o **SAFER+**. Ambos são cifras de blocos de chave simétrica tradicional. O SAFER+ foi submetido aos rígidos testes de aprovação do AES, mas foi eliminado na primeira rodada, porque era mais lento que os outros candidatos. O Bluetooth ficou pronto antes de ser escolhida a cifra do AES; caso contrário, é mais provável que ele tivesse usado o Rijndael.

A criptografia real usando a cifra de fluxo é mostrada na Figura 8.14, com o texto simples sendo submetido a um XOR com o fluxo de chaves para gerar o texto cifrado. Infelizmente, o próprio E_0 (como o RC4) pode ter deficiências fatais (Jakobsson e Wetzel, 2001). Embora não tenha sido rompido na época em que escrevemos, sua semelhança com a cifra A5/1, cuja falha espetacular compromete todo o tráfego de telefones GSM, causa preocupação (Biryukov *et al.*, 2000). Às vezes, é espantoso perceber (até mesmo para este autor) que, no eterno jogo de gato e rato entre criptógrafos e criptoanalistas, os criptoanalistas freqüentemente sejam os vencedores.

Outra questão de segurança é que o Bluetooth efetua a autenticação apenas de dispositivos, não de usuários; assim, o furto de um dispositivo Bluetooth pode dar ao ladrão acesso às finanças e às outras contas do usuário. No entanto, o Bluetooth também implementa segurança nas camadas superiores. Portanto, até

mesmo na eventualidade de uma violação da segurança no nível de enlace, deve restar alguma segurança, especialmente para aplicações que exigem a digitação de um código PIN em algum tipo de teclado para completar a transação.

[T4] Segurança do WAP 2.0

O fórum WAP aprendeu a lição por utilizar uma pilha de protocolos não padronizada no WAP 1.0. Grande parte do WAP 2.0 utiliza protocolos padrão em todas as camadas. A segurança não é exceção. Tendo em vista que ele se baseia no IP, o WAP admite o uso total do IPsec na camada de rede. Na camada de transporte, as conexões TCP podem ser protegidas pelo TLS, um padrão da IETF que estudaremos mais adiante neste capítulo. Em um nível ainda mais alto, ele utiliza a autenticação de clientes HTTP, definida na RFC 2617. As bibliotecas de criptografia da camada de aplicação proporcionam controle de integridade e não repúdio. De modo geral, como o WAP 2.0 se baseia em padrões conhecidos, existe uma chance de que seus serviços de segurança, em particular, privacidade, autenticação, controle de integridade e não repúdio possam ser melhores que a segurança do 802.11 e do Bluetooth.

[T2] 8.7 Protocolos de autenticação

A **autenticação** é a técnica através da qual um processo confirma que seu parceiro na comunicação é quem deve ser e não um impostor. Confirmar a identidade de um processo remoto, face à presença de um intruso ativo mal-intencionado, é surpreendentemente difícil e exige protocolos complexos baseados no uso da criptografia. Nesta seção, estudaremos alguns dos muitos protocolos de autenticação usados em redes de computadores com falhas na segurança.
Por outro lado, algumas pessoas confundem autorização com autenticação. A autenticação lida com a questão de determinar se você está ou não se

comunicando com um processo específico. A autorização se preocupa com o que esse processo tem permissão para fazer. Por exemplo, um processo cliente entra em contato com um servidor de arquivos e afirma: "Sou o processo do Scott e quero excluir o arquivo *cookbook.old*". Do ponto de vista do servidor de arquivos, as seguintes perguntas devem ser respondidas:

1. Esse processo é realmente de Scott (autenticação)?
2. Scott tem permissão para excluir *cookbook.old* (autorização)?

Somente depois que ambas as perguntas forem respondidas afirmativamente sem qualquer ambigüidade, a ação solicitada poderá ser executada. Na verdade, a primeira pergunta é a mais importante. Depois que o servidor de arquivos sabe com quem está se comunicando, verificar a autorização é simplesmente uma questão de pesquisar entradas de tabelas ou bancos de dados locais. Por essa razão, nesta seção vamos nos concentrar na questão da autenticação.

O modelo genérico que todos os protocolos de autenticação utilizam é descrito a seguir. Alice começa enviando uma mensagem para Bob ou para um **KDC (Key Distribution Center)** no qual confia e que sempre é honesto. Acontecem muitas outras trocas de mensagens em diferentes sentidos. À medida que essas mensagens são enviadas, uma intrusa mal-intencionada, Trudy, pode interceptar, modificar ou reproduzir essas mensagens a fim de enganar Alice e Bob, ou simplesmente para atrapalhar.

Todavia, quando a execução do protocolo tiver sido concluída, Alice terá certeza de que está se comunicando com Bob e vice-versa. Além disso, na maioria dos protocolos, os dois também terão estabelecido uma **chave de sessão** secreta que deverá ser usada durante a conversação. Na prática, por motivos de desempenho, todo o tráfego de dados é criptografado utilizando-se a criptografia de chave simétrica (em geral, AES ou DES triplo), embora a criptografia de chave pública seja extensamente usada nos próprios protocolos de autenticação e para

estabelecer a chave de sessão.

O objetivo de se utilizar uma nova chave de sessão escolhida aleatoriamente para cada nova conexão é minimizar o volume de tráfego provocado pelo envio das chaves secretas ou públicas dos usuários, reduzir o volume de texto cifrado que um intruso pode obter e minimizar os danos causados, caso haja uma pane em um processo e seu dump de memória caia em mãos erradas. É muito provável que a única chave presente seja a chave de sessão. Todas as chaves permanentes deverão ser cuidadosamente zeradas depois que a sessão for estabelecida.

[T3] 8.7.1 Autenticação baseada em uma chave secreta compartilhada

Para o nosso primeiro protocolo de autenticação, vamos supor que Alice e Bob já compartilham uma chave secreta, K_{AB}. Essa chave compartilhada pode ter sido definida pelos dois em uma conversa telefônica ou pessoalmente, mas não na rede, que apresenta problemas de segurança.

Esse protocolo se baseia em um princípio encontrado em muitos protocolos de autenticação: um dos lados envia um número aleatório ao outro, que em seguida o transforma de algum modo especial e retorna o resultado. Tais protocolos são chamados protocolos de **desafio-resposta.** Nesse e nos próximos protocolos de autenticação, será usada a seguinte notação:

A e B são as identidades de Alice e Bob

R_i são desafios, sendo que o caractere subscrito identifica o desafiante

K_i são chaves, onde i indica o proprietário; K_S é a chave da sessão

A seqüência de mensagens do nosso primeiro protocolo de autenticação de chave compartilhada é mostrada na Figura 8.32. Na mensagem 1, Alice envia sua identidade A para Bob, de uma forma que Bob possa entender. É claro que Bob não tem como saber se essa mensagem veio de Alice ou de Trudy; portanto, ele escolhe um desafio, um número aleatório muito extenso, R_B, e o envia de volta a

"Alice" como sua mensagem número 2 em texto simples. Os números aleatórios usados apenas uma vez em protocolos de desafio-resposta como esse são chamados **nonces**. Em seguida, Alice criptografa a mensagem com a chave que compartilha com Bob e envia o texto cifrado, $K_{AB}(R_B)$, de volta na mensagem 3. Quando vê a mensagem, Bob fica sabendo imediatamente que ela veio de Alice, pois Trudy não conhece K_{AB} e, portanto, não poderia tê-la gerado. Além disso, como o número R_B foi escolhido aleatoriamente a partir de um espaço muito extenso (digamos, números aleatórios de 128 bits), é muito improvável que Trudy tenha visto R_B e sua resposta em uma sessão anterior. É igualmente improvável que ela conseguisse adivinhar a resposta correta a qualquer desafio.

[arte: ver original p. 787a]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.32

[FL] Uma autenticação bidirecional utilizando um protocolo de desafio-resposta

A essa altura, Bob tem certeza de que está se comunicando com Alice, mas Alice não tem certeza de nada, pois sabe que Trudy pode ter interceptado a mensagem 1 e enviado R_B como resposta. Talvez Bob tenha morrido na noite passada. Para descobrir com quem está se comunicando, Alice seleciona um número aleatório, R_A, e envia a Bob como texto simples, na mensagem 4. Quando Bob responde com $K_{AB}(R_A)$, Alice fica sabendo que está se comunicando com Bob. Se eles quiserem estabelecer uma chave de sessão agora, Alice poderá selecionar uma, K_S, e enviá-la a Bob criptografada com K_{AB}.

O protocolo da Figura 8.32 contém cinco mensagens. Vamos ver se podemos eliminar algumas delas. Uma abordagem é ilustra na Figura 8.33. Nessa figura,

Alice inicia o protocolo de desafio-resposta em vez de esperar que Bob o faça. Da mesma forma, enquanto está respondendo ao desafio de Alice, Bob envia o dele: o protocolo inteiro pode ser reduzido a três mensagens, em vez de cinco.

[arte: ver original p. 787b]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.33

[FL] Um protocolo de autenticação bidirecional abreviado

Esse novo protocolo representa um aperfeiçoamento em relação ao original? Em um certo sentido, isso é verdade, pois agora o protocolo está mais curto. Infelizmente, agora ele também está errado. Sob determinadas circunstâncias, Trudy é capaz de enganar esse protocolo utilizando o método conhecido como **ataque por reflexão.** Especificamente, Trudy poderá rompê-lo se for possível abrir várias sessões com Bob ao mesmo tempo. Por exemplo, essa situação seria verdadeira se Bob fosse um banco e estivesse preparado para aceitar muitas conexões simultâneas enviadas por caixas eletrônicos ao mesmo tempo.

O ataque por reflexão de Trudy é mostrado na Figura 8.34. Ele começa com Trudy afirmando ser Alice e enviando R_T. Bob responde, como sempre, com seu próprio desafio, R_B. Agora Trudy está em apuros. O que ela pode fazer? Ela não conhece $K_{AB}(R_B)$.

[arte: ver original p. 788]

[Dísticos]

[1] Primeira sessão

[2] Segunda sessão

[3] Primeira sessão

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.34

[FL] O ataque por reflexão

Ela pode abrir uma outra sessão com a mensagem 3, fornecendo o R_B extraído da mensagem 2 como seu desafio. Bob o criptografa calmamente e envia $K_{AB}(R_B)$ na mensagem 4. Representamos com sombreados as mensagens da segunda sessão, a fim de destacá-las. Agora, Trudy tem as informações que faltavam e, portanto, pode concluir a primeira sessão e abandonar a segunda. Nesse momento, Bob está convencido de que Trudy é Alice e, quando ela pede o saldo da conta, ele o informa sem maiores perguntas. Em seguida, quando Trudy pede que Bob transfira todo o dinheiro para uma conta secreta na Suíça, ele não hesita em fazê-lo.

A moral da história é a seguinte:

Projetar um protocolo de autenticação correto é mais difícil do que parece.

Com freqüência, quatro regras podem nos ajudar nesse caso. Elas são as seguintes:

1. Fazer com que o transmissor prove quem é antes do receptor responder. Nesse caso, Bob revela informações valiosas antes de Trudy fornecer alguma prova de quem é ela.

2. Fazer com que o transmissor e o receptor utilizem chaves específicas para provarem quem são, mesmo que isso signifique ter duas chaves compartilhadas, K_{AB} e K'_{AB}.

3. Fazer com que o transmissor e o receptor extraiam seus desafios de conjuntos distintos. Por exemplo, o transmissor deve usar números pares e o transmissor

deve usar números ímpares.

4. Tornar o protocolo resistente a ataques que envolvam uma segunda sessão paralela, no qual as informações obtidas em uma sessão sejam usadas em uma sessão diferente.

Se até mesmo uma dessas regras for violada, isso significa que o protocolo poderá ser violado com freqüência. Nesse caso, todas as quatro regras foram violadas, com conseqüências desastrosas.

Agora examinar mais de perto a Figura 8.32. É possível garantir que esse protocolo não está sujeito a um ataque por reflexão? Bem, depende. Essa é uma questão bastante sutil. Trudy foi capaz de violar nosso protocolo usando um ataque por reflexão, porque foi possível abrir uma segunda sessão com Bob e enganá-lo, respondendo a suas próprias perguntas. O que aconteceria se Alice fosse um computador de uso geral que também aceitasse várias sessões, em vez de ser uma pessoa diante de um computador? Vejamos o que Trudy pode fazer nesse caso.

Para ver como funciona o ataque de Trudy, observe a Figura 8.35. Alice começa anunciando sua identidade na mensagem 1. Trudy intercepta essa mensagem e inicia sua própria sessão com a mensagem 2, afirmando ser Bob. Novamente sombreamos as mensagens da sessão 2. Alice responde à mensagem 2 com a mensagem 3, dizendo: "Você afirma ser Bob? Então, prove". Nesse momento Trudy não em saída, porque não pode provar ser Bob.

[arte: ver original p. 789]

[Dísticos]

[1] Primeira sessão

[2] Segunda sessão

[3] Primeira sessão

[4] Segunda sessão

[5] Primeira sessão

[6] Segunda sessão

[7] Primeira sessão

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.35

[FL] Um ataque por reflexão no protocolo da Figura 8.32

O que Trudy faz agora? Ela volta para a primeira sessão, onde é sua vez de enviar um desafio, e transmite a R_A que obteve na mensagem 3. Alice amavelmente responde na mensagem 5 e, desse modo, fornece a Trudy as informações de que ela precisa para enviar a mensagem 6 na sessão 2. Nesse momento, Trudy está à vontade, porque conseguiu responder com sucesso ao desafio de Alice na sessão 2. Agora ela pode cancelar a sessão 1, transmitir qualquer número antigo para o restante da sessão 2, e terá uma sessão autenticada com Alice na sessão 2. Contudo, Trudy é malvada e quer causar ainda mais danos. Em vez de enviar qualquer número antigo para concluir a sessão 2, ela espera até Alice enviar a mensagem 7, o desafio de Alice para a sessão 1. É claro que Trudy não sabe como responder, e portanto utiliza outra vez o ataque por reflexão, devolvendo R_{A2} como a mensagem 8. Alice codifica R_{A2} de maneira conveniente na mensagem 9. Agora, Trudy volta para a sessão 1 e envia a Alice, na mensagem 10, o número que ela deseja, cuidadosamente copiado do número que a própria Alice enviou na mensagem 9. Nesse momento, Trudy tem duas sessões completamente autenticadas com Alice.

Esse ataque tem um resultado um pouco diferente do ataque no protocolo de três mensagens, ilustrado na Figura 8.34. Dessa vez, Trudy tem duas conexões

autenticadas com Alice. No exemplo anterior, ela tinha uma conexão autenticada com Bob. Mais uma vez, se aplicássemos todas as regras gerais de protocolos de autenticação discutidas antes, esse ataque poderia ter sido interrompido. Uma descrição detalhada desses tipos de ataques e de como frustrá-los é apresentada em (Bird *et al.*, 1993), que também mostram como é possível construir de forma sistemática protocolos que comprovadamente são corretos. Porém, o mais simples desses protocolos é um pouco complicado; portanto, mostraremos agora uma classe diferente de protocolos que também funcionam.

O novo protocolo de autenticação é mostrado na Figura 8.36 (Bird *et al.*, 1993). Ele emprega um HMAC do tipo que vimos quando estudamos o IPsec. Alice começa enviando a Bob um nonce R_A como mensagem 1. Bob responde selecionando seu próprio nonce, R_B, e devolvendo-o juntamente com um HMAC. O HMAC é formado com o objetivo de construir uma estrutura de dados que consiste no nonce de Alice, no nonce de Bob, em suas identidades e na chave secreta compartilhada, K_{AB}. Esses dados estruturados passam então por um hash no HMAC, por exemplo usando SHA-1. Quando receber a mensagem 2, Alice terá R_A (que ela própria escolheu), R_B, que chegará sob a forma de texto simples, as duas identidades e a chave secreta K_{AB}, conhecida desde o início, e depois ela mesma poderá calcular o HMAC. Se este corresponder ao HMAC da mensagem, ela saberá que está se comunicando com Bob, porque Trudy não conhece K_{AB} e, desse modo, não terá como saber qual HMAC enviar. Alice responde a Bob com um HMAC contendo apenas os dois nonces.

[arte: ver original p. 790]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.36

[FL] Autenticação usando HMACs

Trudy pode subverter de algum modo esse protocolo? Não, porque ela não é capaz de forçar qualquer das partes a codificar ou fazer o hash de um valor de sua escolha, como aconteceu na Figura 8.34 e na Figura 8.(Fo [Não conseguimos definir se essa citação se refere à Figura 8.35 ou 8.36. Favor verificar com a editora do OA]. Ambos os HMACs incluem valores escolhidos pela parte transmissora, algo que Trudy não pode controlar.

Utilizar HMACs não é o único meio de se empregar essa idéia. Um esquema alternativo usado com freqüência em vez de calcular o HMAC sobre uma série de itens é codificar os itens seqüencialmente usando o encadeamento de blocos de cifras.

[T3] 8.7.2 Como estabelecer uma chave compartilhada: A troca de chaves de Diffie-Hellman

Até agora partimos do princípio de que Alice e Bob compartilham uma chave secreta. Suponha que isso não seja verdade (porque até agora não há nenhuma PKI universalmente aceita para assinar e distribuir certificados). Como eles podem estabelecer uma chave secreta? Uma possibilidade seria Alice telefonar para Bob e dar sua chave a ele, mas provavelmente ele começaria a conversa dizendo: "Como posso saber que você é Alice e não Trudy?" Eles poderiam tentar se encontrar pessoalmente, com cada um levando passaporte, carteira de motorista e três cartões de crédito. No entanto, como são muito ocupados, talvez não consigam encontrar uma data conveniente para ambos durante meses. Felizmente, apesar de parecer incrível, há uma forma de pessoas que não se conhecem estabelecerem uma chave secreta em plena luz do dia, mesmo com Trudy registrando cuidadosamente cada mensagem.

O protocolo que permite o estabelecimento de uma chave secreta entre pessoas que não se conhecem é chamado **troca de chaves de Diffie-Hellman** (Diffie e Hellman, 1976) e funciona da forma descrita a seguir. Alice e Bob têm de concordar em relação a dois números extensos, n e g, onde n é um número primo, $(n - 1)/2$ também é um número primo e onde certas condições se aplicam a g. Esses números podem ser públicos; portanto, um deles só precisa selecionar n e g e informar ao outro abertamente. Agora, Alice escolhe um grande número x (digamos, de 512 bits) e o mantém em segredo. Da mesma forma, Bob seleciona um grande número secreto, y.

Alice inicia um protocolo de troca de chaves enviando a Bob uma mensagem contendo (n, g, $g^x \bmod n$), como mostra a Figura 8.37. Bob responde enviando a Alice uma mensagem contendo $g^y \bmod n$. Agora, Alice eleva a x-ésima potência em módulo n o número que Bob lhe enviou, a fim de obter $(g^y \bmod n)^x \bmod n$. Bob executa uma operação semelhante para obter $(g^x \bmod n)^y \bmod n$. Pelas leis da aritmética modular, ambos os cálculos produzem $g^{xy} \bmod n$. Agora, Alice e Bob compartilham uma chave secreta, $g^{xy} \bmod n$.

[arte: ver original p. 791]

[Dísticos]

[1] Alice escolhe x

[2] Bob escolhe y

[3] Alice calcula $(g^y \bmod n)^x \bmod n = g^{xy} \bmod n$

[4] Bob calcula $(g^x \bmod n)^y \bmod n = g^{xy} \bmod n$

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.37

[FL] A troca de chaves de Diffie-Hellman

É óbvio que Trudy viu as duas mensagens. Com base na mensagem 1, ela conhece g e n. Se pudesse calcular x e y, Trudy poderia descobrir a chave secreta. O problema é que com apenas g^x mod n, ela não consegue encontrar x. Não existem algoritmos práticos para o cálculo de logaritmos discretos cuja base é um número primo muito extenso conhecido.

Para tornar o exemplo mostrado antes mais concreto, utilizaremos os valores (completamente falsos): $n = 47$ e $g = 3$. Alice seleciona $x = 8$ e Bob seleciona $y = 10$, o que é mantido em segredo. A mensagem de Alice para Bob é (47, 3, 28), pois 3^8 mod 47 é igual a 28. A mensagem de Bob para Alice é (17). Alice calcula 17^8 mod 47, que é igual a 4. Bob calcula 28^{10} mod 47, que é igual a 4. Alice e Bob determinaram de forma independente que agora a chave secreta é 4. Trudy tem de resolver a equação 3^x mod $47 = 28$, o que pode ser feito por pesquisa exaustiva de números pequenos como esse, mas não quando todos os números têm centenas de bits. Todos os algoritmos conhecidos até o momento demoram muito tempo para realizar esse cálculo, mesmo em supercomputadores maciçamente paralelos.

Apesar da elegância do algoritmo de Diffie-Hellman, há um problema: quando Bob obtém a tripla (47, 3, 28), como ele pode saber se ela veio de Alice e não de Trudy? A resposta é simples: ele não tem como saber. Infelizmente, Trudy pode explorar esse fato para enganar Alice e Bob, como ilustra a Figura 8.38. Aqui, enquanto Alice e Bob estão escolhendo x e y, respectivamente, Trudy escolhe seu próprio número aleatório, z. Alice envia a mensagem 1 para Bob. Trudy a intercepta e envia a mensagem 2 para Bob, usando g e n corretos (que, de qualquer forma, são públicos), mas com seu próprio z em vez de x. Ela também envia a mensagem 3 de volta para Alice. Mais tarde, Bob envia a mensagem 4 para Alice, que Trudy mais uma vez intercepta e guarda.

Agora, todos utilizam a aritmética modular. Alice calcula a chave secreta como g^{xz} mod n, e Trudy faz o mesmo (para mensagens enviadas a Alice). Bob calcula g^{yz} mod n e Trudy faz o mesmo (nas mensagens enviadas a Bob). Alice pensa que está se comunicando com Bob e, portanto, estabelece uma chave de sessão (com Trudy). Bob faz o mesmo. Todas as mensagens que Alice envia na sessão criptografada são capturadas, armazenadas e modificadas por Trudy para então (opcionalmente) serem passadas a Bob. Da mesma forma, no outro sentido, Trudy vê tudo e pode modificar todas as mensagens como quiser, enquanto Alice e Bob têm a ilusão de que há um canal seguro entre os dois. Esse método é chamado **ataque da brigada de incêndio**, pois lembra vagamente um antigo corpo de bombeiros formado por voluntários que, enfileirados, passavam baldes d'água de mão em mão do caminhão até o incêndio. Esse método também é chamado ataque de **@@@homem em posição intermediária**.

[arte: ver original p. 792]

[Dísticos]

[1] Alice escolhe x

[2] Trudy escolhe z

[3] Bob escolhe y

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.38

[FL] O ataque de brigada de incêndio ou de homem em posição intermediária

[T3] 8.7.3 Autenticação com o uso de um centro de distribuição de chaves

A idéia de compartilhar um segredo com uma pessoa estranha quase funcionou. Por outro lado, talvez não tenha valido a pena a tentativa (ataque da raposa e das

uvas). Para conversar com *n* pessoas dessa forma, você precisará de *n* chaves. Para pessoas famosas, o gerenciamento de chaves poderia se tornar uma grande dor de cabeça, especialmente se cada chave tivesse de ser armazenada em uma placa de chips plásticos separada.

Uma outra estratégia é introduzir um centro de distribuição de chaves (KDC — key distribution center) confiável. Nesse modelo, cada usuário tem uma única chave compartilhada com o KDC. Agora o gerenciamento de sessão e de autenticação passa pelo KDC. O protocolo de autenticação para o KDC mais simples envolve duas partes e um KDC confiável e é descrito na Figura 8.39.

[arte: ver original p. 793]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.39

[FL] Uma primeira tentativa de protocolo de autenticação usando um KDC

A idéia por trás desse protocolo é simples: Alice escolhe uma chave de sessão K_S e informa ao KDC que deseja se comunicar com Bob usando K_S. Essa mensagem é criptografada com a chave secreta que Alice compartilha (apenas) com o KDC, K_A. O KDC descriptografa essa mensagem, extraindo a identidade de Bob e a chave de sessão. Em seguida, cria uma nova mensagem contendo a identidade de Alice e a chave de sessão, e depois envia essa mensagem a Bob. Essa criptografia é feita com K_B, a chave secreta que Bob compartilha com o KDC. Quando descriptografa a mensagem, Bob fica sabendo que Alice quer se comunicar com ele e qual chave ela desejar usar.

Aqui, a autenticação ocorre sem maiores problemas. O KDC sabe que a mensagem 1 deve vir de Alice, pois ninguém mais teria sido capaz de

criptografá-la com a chave secreta de Alice. Da mesma forma, Bob sabe que a mensagem 2 deve ter vindo do KDC, em quem ele confia, pois ninguém mais conhece sua chave secreta.

Infelizmente, esse protocolo tem uma falha muito séria. Trudy precisa de dinheiro; portanto, ela pensa em alguns serviços legítimos que pode fazer para Alice, faz uma oferta interessante e consegue o trabalho. Depois de fazer o serviço, Trudy, com toda educação, solicita que Alice faça o pagamento através de transferência bancária. Em seguida, Alice estabelece uma chave de sessão com o funcionário do banco, Bob. Em seguida, ela envia a Bob uma mensagem solicitando que o dinheiro seja transferido para a conta de Trudy.

Nesse ínterim, Trudy volta a bisbilhotar a rede. Ela copia tanto a mensagem 2 da Figura 8.39 quanto a solicitação de transferência de dinheiro enviada depois da mensagem. Posteriormente, ela responde a ambas as mensagens de Bob. Bob as recebe e pensa: "Alice deve ter contratado Trudy outra vez. Percebe-se que o trabalho dela é muito bom". Em seguida, Bob transfere uma quantia igual em dinheiro da conta de Alice para a conta de Trudy. Algum tempo depois do 50º par de mensagens, Bob vai até Trudy e oferece um bom empréstimo para que ela possa expandir seus negócios, que obviamente vão muito bem. Esse problema é chamado **ataque por repetição**.

Existem várias soluções para o ataque por repetição. A primeira é incluir um timbre de hora em cada mensagem. Em seguida, se alguém receber uma mensagem obsoleta, ela poderá ser descartada. O problema dessa estratégia é que os relógios nunca estão sincronizados com exatidão na rede; portanto, deve haver um período durante o qual esse timbre de hora será válido. Trudy pode repetir a mensagem durante esse período e se livrar dela.

A segunda solução é colocar um nonce em cada mensagem. Nesse caso, cada parte terá de se lembrar de todos os nonces anteriores e rejeitar as mensagens

que contenham um nonce já utilizado. No entanto, os nonces têm de ser memorizados para sempre, a menos que Trudy tente repetir uma mensagem com 5 anos de existência. Além disso, se alguma máquina apresentar falha e perder sua lista de nonces, ela estará vulnerável a um ataque por repetição. Os timbres de hora e os nonces podem ser combinados para limitar o tempo durante o qual os nonces têm de ser memorizados, mas é óbvio que o protocolo ficará muito mais complicado.

Um enfoque mais sofisticado para a autenticação mútua é usar um protocolo de desafio-resposta que funcione em diversas direções. Um exemplo bastante conhecido desse tipo de protocolo é o protocolo de **autenticação de Needham-Schroeder** (Needham e Schroeder, 1978), sendo que uma de suas variantes é mostrada na Figura 8.40.

[arte: ver original p. 794]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.40

[FL] O protocolo de autenticação Needham-Schroeder

O protocolo começa com Alice informando ao KDC que deseja se comunicar com Bob. Essa mensagem contém um número extenso aleatório, R_A, que é usado como nonce. O KDC retorna a mensagem 2 contendo o número aleatório de Alice, uma chave de sessão e um bilhete que ela pode enviar a Bob. O objetivo do número aleatório, R_A, é garantir a Alice que a mensagem 2 é nova e não uma repetição. A identidade de Bob também é enviada, caso Trudy pense na possibilidade de substituir *B* na mensagem 1 por sua própria identidade, para que o KDC codifique o bilhete no fim da mensagem 2 com K_T em vez de K_B. O bilhete codificado com

K_B é incluído na mensagem criptografada para impedir que Trudy o substitua por algo diferente quando ele retornar a Alice.

Agora Alice envia o bilhete a Bob, junto com um novo número aleatório, R_{A2}, criptografado com a chave de sessão, K_S. Na mensagem 4, Bob envia $K_S(R_{A2} - 1)$, a fim de provar a Alice que ela está se comunicando com o verdadeiro Bob. Retornar $K_S(R_{A2})$ não seria uma boa opção, pois Trudy poderia ter acabado de roubar essa chave na mensagem 3.

Depois de receber a mensagem 4, Alice estará convencida de que está se comunicando com Bob e de que nenhuma repetição poderia ter sido usada até agora. Afinal de contas, ela acabou de gerar R_{A2} há alguns milissegundos. O objetivo da mensagem 5 é convencer Bob de que ele está se comunicando realmente com Alice, e que nenhuma repetição está sendo usada aqui. Ao fazer com que cada parte gere um desafio e responda a outro, a possibilidade de qualquer tipo de ataque por repetição é eliminada.

Apesar de parecer bastante sólido, esse protocolo tem uma pequena falha. Se Trudy conseguir obter uma antiga chave de sessão no texto simples, ela poderá dar início a uma nova sessão com Bob repetindo a mensagem 3 correspondente à chave comprometida e convencê-lo de que é Alice (Denning e Sacco, 1981). Dessa vez, ela poderá desfalcar a conta bancária de Alice sem precisar prestar qualquer serviço honesto.

Mais tarde, Needham e Schroeder publicaram um protocolo que corrige esse problema (Needham e Schroeder, 1987). No mesmo exemplar do mesmo jornal, Otway e Rees (1987) também publicaram um protocolo que resolve o problema de uma forma mais simples. A Figura 8.41 mostra um protocolo de Otway-Rees ligeiramente modificado.

[arte: ver original p. 795]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.41

[FL] O protocolo de autenticação de Otway-Rees (ligeiramente simplificado)

No protocolo de Otway-Rees, Alice começa gerando um par de números aleatórios, R, que será usado como um identificador comum, R_A, que Alice utilizará para desafiar Bob. Quando receber essa mensagem, Bob criará uma nova mensagem com a parte criptografada da mensagem de Alice e uma outra mensagem análoga de sua própria autoria. Ambas as partes criptografadas com K_A e K_B identificam Alice e Bob, e contêm o identificador e um desafio.

O KDC verifica se o R de ambas as partes é igual. Talvez não seja, porque Trudy adulterou R na mensagem 1 ou substituiu parte da mensagem 2. Se os dois números R coincidirem, o KDC considerará a mensagem de solicitação de Bob válida. Em seguida, o KDC irá gerar uma chave de sessão, criptografada duas vezes, uma para Alice e outra para Bob. Cada mensagem conterá o número aleatório do receptor, como prova de que o KDC, e não Trudy, gerou a mensagem. Nesse momento, tanto Alice quanto Bob têm a posse da mesma chave de sessão e podem dar início à comunicação. Na primeira vez que eles trocarem mensagens de dados, cada um poderá ver que o outro tem uma cópia idêntica de K_S, e assim a autenticação é concluída.

[T3] 8.7.4 Autenticação com a utilização do Kerberos

Um protocolo de autenticação usado em muitos sistemas reais (inclusive no Windows 2000) é o **Kerberos**, que se baseia em uma variante do protocolo de Needham-Schroeder. Seu nome se deve ao cão de várias cabeças da mitologia grega que guardava a entrada para Hades (provavelmente para manter as pessoas

indesejáveis à distância). O Kerberos foi projetado no MIT para permitir que os usuários de estações de trabalho tivessem acesso a recursos da rede de uma forma segura. Sua grande diferença em relação ao protocolo de Needham-Schroeder é sua suposição de que todos os clocks estão muito bem sincronizados. O protocolo passou por várias iterações. A V4 é a versão mais usada na indústria; sendo assim, vamos descrevê-la. Depois disso, faremos alguns comentários sobre a versão seguinte, V5. Para obter mais informações, consulte (Steiner *et al.*, 1988).

O Kerberos envolve três servidores além de Alice (uma estação de trabalho cliente):

O Authentication Server (AS): confirma a identidade dos usuários durante o processo de login.

O Ticket-Granting Server (TGS): emite "bilhetes de comprovação de identidade".

O servidor de arquivos Bob: na verdade, faz o trabalho que Alice deseja ver pronto.

O AS é semelhante a um KDC, porque compartilha uma senha secreta com todos os usuários. O trabalho do TGS é emitir bilhetes que possam convencer os servidores reais de que o portador de um bilhete TGS realmente é quem afirma ser.

Para dar início a uma sessão, Alice utiliza uma estação de trabalho arbitrária e digita seu nome. A estação de trabalho envia seu nome ao AS em texto simples, como mostra a Figura 8.42. O que retorna é uma chave de sessão e um bilhete, $K_{TGS}(A, K_S)$, destinado ao TGS. Esses itens são empacotados juntos e criptografados com a chave secreta de Alice, de modo que apenas Alice seja capaz de descriptografá-los. Somente quando a mensagem 2 chega, a estação de trabalho pede a senha de Alice. Em seguida, a senha é usada para gerar K_A, a fim

de descriptografar a mensagem 2 e obter a chave de sessão e o bilhete TGS que ela contém. Nesse momento, a estação de trabalho substitui a senha de Alice, para garantir que a senha só estará na estação de trabalho durante alguns milissegundos, no máximo. Se Trudy tentar estabelecer um login como Alice, a senha que ela digitar estará errada e a estação de trabalho detectará o problema, porque o trecho padrão da mensagem 2 estará incorreto.

Depois de estabelecer o login, Alice pode informar à estação de trabalho que deseja entrar em contato com Bob, o servidor de arquivos. Em seguida, a estação de trabalho envia a mensagem 3 ao TGS solicitando um bilhete para usar com Bob. O principal elemento nessa solicitação é $K_{TGS}(A, K_S)$, que é criptografado com a chave secreta de TGS e é usado como prova de que o transmissor realmente é Alice. O TGS responde criando uma chave de sessão, K_{AB}, para que Alice a utilize com Bob. Duas versões dessa chave são retornadas. A primeira é criptografada apenas com K_S, para que Alice possa ler a mensagem. A segunda é criptografada com a chave de Bob, K_B, de forma que Bob também possa ler a mensagem.

[arte: ver original p. 797]

[Dísticos]

[1] Login

[2] Obtém um bilhete

[3] Executa o trabalho

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.42

[FL] A operação do Kerberos V4

Trudy pode copiar a mensagem 3 e tentar usá-la mais uma vez, mas será

frustrada pelo timbre de hora criptografado, *t*, enviado junto com a mensagem. Trudy não pode substituir o timbre de hora por outro mais recente, porque não conhece K_S, a chave de sessão que Alice utiliza para se comunicar com o TGS. Mesmo que Trudy repita a mensagem 3 rapidamente, tudo o que ela obterá será outra cópia da mensagem 4, que não pôde descriptografar antes e que também não poderá descriptografar agora.

Nesse momento, Alice envia K_{AB} a Bob, com a finalidade de estabelecer uma sessão com ele. Essa troca também recebe um timbre de hora. A resposta é a prova de que Alice está realmente se comunicando com Bob, e não com Trudy.

Depois de uma série de trocas, Alice pode se comunicar com Bob sob a proteção de K_{AB}. Se mais tarde ela chegar à conclusão de que precisa se comunicar com outro servidor, Carol, Alice simplesmente repetirá a mensagem 3 para o TGS, apenas especificando *C* em vez de *B*. O TGS responderá prontamente com um bilhete criptografado com K_C, que Alice poderá enviar a Carol e que Carol aceitará como prova de que Alice o enviou.

O objetivo de todo esse trabalho é que agora Alice pode acessar servidores instalados por toda a rede de forma segura, e sua senha nunca terá de percorrer a rede. Na verdade, a senha só precisaria permanecer na estação de trabalho da própria Alice durante alguns milissegundos. No entanto, observe que cada servidor faz sua própria autorização. Quando Alice apresenta seu bilhete a Bob, isso prova a Bob quem o enviou. Na verdade, a decisão sobre o que Alice está autorizada a fazer cabe a Bob.

Como os projetistas do Kerberos não esperavam que o mundo inteiro confiasse em um único servidor de autenticação, eles reservaram espaço para o uso de diversos **domínios (realms)**, cada um com seu próprio AS e TGS. Para obter um bilhete referente a um servidor de um domínio distante, Alice solicitaria a seu próprio TGS um bilhete aceito pelo TGS do domínio distante. Se o TGS distante

tiver sido registrado com o TGS local (exatamente como fazem os servidores locais), o TGS local dará a Alice um bilhete válido no TGS distante. Depois disso, ela poderá usá-lo para executar determinadas ações, como obter bilhetes para os servidores desse domínio. No entanto observe que, para partes pertencentes a dois domínios interagirem, cada uma delas deverá confiar no TGS da outra.
O Kerberos V5 tem mais recursos do que o V4 e um overhead maior. Ele também utiliza a ASN.1 (Abstract Syntax Notation 1) da OSI para descrever tipos de dados e tem pequenas alterações nos protocolos. Além disso, o tempo de duração de seus bilhetes é mais longo, ele permite que os bilhetes sejam renovados e emite bilhetes pós-datados. Além disso, pelo menos na teoria, ele não é dependente do DES, como é o caso do V4, e aceita vários domínios, delegando a geração de bilhetes a diversos servidores de bilhetes.

[T3] 8.7.5 Autenticação com a criptografia de chave pública

Também é possível fazer uma autenticação mútua através do uso da criptografia de chave pública. Para começar, Alice precisa da chave pública de Bob. Se existir uma PKI com o servidor de diretórios que entregue certificados para chaves públicas, Alice poderá solicitar o certificado de Bob, como mostra a Figura 8.43 na mensagem 1. A resposta na mensagem 2 é um certificado X.509 que contém a chave pública de Bob. Quando Alice verifica que a assinatura está correta, ela envia a Bob uma mensagem contendo sua identidade e um nonce.

[arte: ver original p. 798]

[Dísticos]

[1] Diretório

[2]

1. Forneça E_B

2. Aqui está E_B

[3]

4. Forneça E_A

5. Aqui está E_A

Atenção, produção!

Não foi possível reproduzir os outros dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há mais nada a traduzir.

[F]Figura 8.43

[FL] Autenticação mútua com a utilização da criptografia de chave pública

Quando recebe essa mensagem, Bob não tem certeza se ela veio de Alice ou de Trudy, mas continua e pede ao servidor de diretórios a chave pública de Alice (mensagem 4) e logo a recebe (mensagem 5). Em seguida, ele envia a Alice uma mensagem contendo o R_A de Alice, seu próprio nonce, R_B, e uma chave de sessão sugerida, K_S, como a mensagem 6.

Quando recebe a mensagem 6, Alice a descriptografa usando sua própria chave privada. Ela vê R_A na mensagem e fica feliz. A mensagem deve ter vindo de Bob, pois Trudy não tem como determinar R_A. Além disso, a mensagem deve ser nova, e não uma repetição, pois ela acabou de enviar R_A a Bob. Alice concorda com a sessão retornando a mensagem 7. Quando vê R_B criptografada com a chave de sessão que acabou de gerar, Bob fica sabendo que Alice recebeu a mensagem 6 e confirmou R_A.

O que Trudy pode fazer para subverter esse protocolo? Ela pode falsificar a mensagem 3 e enganar Bob fazendo-o pensar que ela é Alice, mas Alice verá um R_A que não enviou e não prosseguirá com a transmissão. Trudy não poderá forjar a mensagem 7 de forma convincente, pois não conhece os valores de R_B e de K_S e não pode determiná-los sem a chave privada de Alice. Ela está sem sorte.

[T2] 8.8 Segurança de correio eletrônico

Quando uma mensagem de correio eletrônico é enviada entre dois sites distantes, geralmente ela transita por dezenas de máquinas até chegar a seu destino. Qualquer uma dessas máquinas pode ler e armazenar a mensagem para usá-la posteriormente. Na prática, não há privacidade, apesar de muita gente achar o contrário. Todavia, muitas pessoas gostariam de enviar mensagens de correio eletrônico para que fossem lidas pelo destinatário pretendido e por ninguém mais (nem seu chefe, nem o governo). Esse desejo estimulou muitas pessoas e grupos a aplicarem os princípios da criptografia que estudamos anteriormente para produzir mensagens seguras. Nas seções a seguir, estudaremos um sistema de correio eletrônico seguro e bastante utilizado, o PGP, e depois mencionaremos brevemente dois outros, o PEM e o S/MIME. Para obter mais informações sobre correio eletrônico seguro, consulte (Kaufman *et al*., 2002; e Schneier, 1995).

[T3] 8.8.1 PGP — Pretty Good Privacy

Basicamente, nosso primeiro exemplo, o **PGP (Pretty Good Privacy)** foi criado por uma única pessoa, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann é um defensor da privacidade cujo lema é: "Se a privacidade for proscrita, somente os proscritos terão privacidade". Lançado em 1991, o PGP é um pacote completo para segurança de mensagens de correio eletrônico que fornece privacidade, autenticação, assinaturas digitais e compactação, tudo de uma forma fácil de usar. Além disso, o pacote completo, incluindo o código-fonte, é distribuído de graça via Internet. Devido à sua qualidade, preço (zero) e disponibilidade em plataformas UNIX, Linux, Windows e Mac OS, esse sistema é extensamente utilizado hoje.

O PGP codifica dados usando uma cifra de bloco chamada **IDEA (International Data Encryption Algorithm)**, que utiliza chaves de 128 bits. Ele foi criado na Suíça

em uma época na qual o DES era considerado decadente e o AES ainda não tinha surgido. Conceitualmente, o IDEA é semelhante ao DES e ao AES: ele mistura os bits em uma série de rodadas, mas os detalhes das funções de mixagem são diferentes do DES e do AES. O gerenciamento de chaves utiliza o RSA e a integridade de dados utiliza o MD5, tópicos que já discutimos.

O PGP também esteve envolvido em controvérsias desde o início (Levy, 1993). Como Zimmermann não fez nada para impedir que outras pessoas colocassem o PGP na Internet, onde gente de todo o mundo poderia obtê-lo, o governo dos Estados Unidos afirmou que Zimmermann violou leis americanas que proibiam a exportação de munições. A investigação que o governo dos EUA fez de Zimmermann durou 5 anos, mas foi abandonada, provavelmente por duas razões. Primeiro, Zimmermann não colocou o PGP na Internet, e assim seu advogado afirmou que *ele* nunca exportou nada (e, na época, havia dúvidas se criar um Web site constituiria uma forma de exportação). Em segundo lugar, o governo percebeu mais tarde que vencer uma disputa judicial significava convencer um júri de que um Web site contendo um programa de privacidade que podia ser transferido por download era uma infração sujeita às penas da lei contra tráfico de armas que proibia a exportação de materiais de guerra como tanques, submarinos, aeronaves militares e armas nucleares. Vários anos de publicidade negativa provavelmente também não ajudariam muito.

A propósito, as regras de exportação são bizarras, para dizer o mínimo. O governo considerava a colocação de código em um Web site um ato de exportação ilegal e processou Zimmermann durante 5 anos. Por outro lado, quando alguém publicava o código-fonte completo do PGP em linguagem C sob a forma de um livro (em uma fonte de tamanho grande com um total de verificação em cada página para facilitar a digitalização) e depois exportava o livro, isso era legal para o governo, porque os livros não são classificados como munições. A

espada é mais poderosa que a caneta, pelo menos para Tio Sam.

Outro problema que o PGP enfrentou envolvia a violação de patentes. A empresa que detinha a patente do RSA, denominada RAS Security, Inc., alegou que o uso que o PGP fazia do algoritmo RSA infringia sua patente, mas esse problema foi contornado nas versões seguintes, a partir da versão 2.6. Além disso, o PGP usa outro algoritmo de criptografia patenteado, o IDEA, cujo uso causou alguns problemas no início.

Tendo em vista que o PGP tem código-fonte aberto, muitas pessoas e grupos o modificaram e produziram várias versões. Algumas delas foram projetadas para contornar as leis de munições, outras se concentraram em evitar o uso de algoritmos patenteados, e ainda outras queriam transformá-lo em um produto comercial com código-fonte fechado. Embora as leis de munições tenham sido ligeiramente atenuadas (do contrário, produtos que usassem o AES não poderiam ter sido exportados pelos Estados Unidos) e a patente do RSA tenha expirado em setembro de 2000, o legado de todos esses problemas foi a existência de várias versões incompatíveis do PGP, identificadas por diversos nomes. A descrição a seguir se concentra no PGP clássico, a versão mais antiga e mais simples. Outra versão popular, o Open PGP, é descrita na RFC 2440. Ainda outra é o Privacy Guard do GNU.

O PGP utiliza intencionalmente algoritmos criptográficos que já existiam em vez de criar novos. Ele se baseia amplamente em algoritmos que passaram por intensas revisões e não foram projetados ou influenciados por qualquer agência governamental que tentasse enfraquecê-los. Para pessoas que tendem a não acreditar no governo, essa característica representa uma excelente opção.

O PGP aceita a compactação de textos, sigilo e assinaturas digitais, e também oferece amplos recursos de gerenciamento de chaves mas, estranhamente, não oferece recursos de correio eletrônico. Ele é mais parecido com um pré-

processador que recebe texto simples como entrada e produz texto cifrado assinado em base64 como saída. Essa saída pode então ser enviada por correio eletrônico. Algumas implementações do PGP chamam um agente do usuário na etapa final para enviar de fato a mensagem.

Para ver como o PGP funciona, considere o exemplo da Figura 8.44. Aqui, Alice deseja enviar uma mensagem em texto simples assinada, *P*, para Bob de forma segura. Tanto Alice quanto Bob têm chaves RSA privadas (D_X) e públicas (E_X). Vamos supor que cada um conheça a chave pública do outro; examinaremos em breve o gerenciamento de chaves do PGP.

[arte: ver original p. 801]

[Dísticos]

[1] K_M : Chave de mensagem de uma única vez para IDEA

[ver símbolo] : Concatenação

[2] Chave privada RSA de Alice, D_A

P MD5 RSA [ver símbolo] P1

[3] Mensagem em texto simples original enviada por Alice

[4] Concatenação de P e do hash assinado de P

[5] Concatenação de P1.Z criptografada com IDEA e K_M criptografada com E_B

[6] Base 64

Texto ASCII para a rede

[7] Chave pública RSA de Bob, E_B

K_M RSA

[8] Zip P1.Z IDEA

P1 compactada

[F]Figura 8.44

[FL] O PGP em operação para enviar uma mensagem

Alice começa invocando o programa PGP em seu computador. Primeiro, o PGP submete sua mensagem *P* a um processo de hash, utilizando o MD5; em seguida, criptografa o resultado empregando sua chave privada RSA, D_A. Quando receber a mensagem, Bob poderá descriptografar o hash com a chave pública de Alice e confirmar que o hash está correto. Mesmo que alguma outra pessoa (por exemplo, Trudy) pudesse adquirir o hash nesse estágio e descriptografá-lo com a chave pública de Alice, a robustez do MD5 garante que seria inviável em termos computacionais produzir outra mensagem com o mesmo hash MD5.

O hash criptografado e a mensagem original são concatenados em uma única mensagem *P*1 e compactados com o programa ZIP, que emprega o algoritmo de Ziv-Lempel (Ziv e Lempel, 1977). Chame a saída dessa etapa de *P1.Z*.

Em seguida, o PGP solicita que Alice informe dados aleatoriamente. O conteúdo e a velocidade de digitação são usados para gerar uma chave IDEA de 128 bits, K_M (denominada chave de sessão na literatura sobre o PGP; no entanto, essa denominação não é adequada, pois não há sessão). Em seguida, K_M é usada para criptografar *P1.Z* com o IDEA no modo de feedback de cifra. Além disso, K_M é criptografada com a chave pública de Bob, E_B. Em seguida, esses dois componentes são concatenados e convertidos para base64, como discutimos na seção sobre o MIME no Capítulo 7. A mensagem resultante contém apenas letras, dígitos e os símbolos +, / e =, o que significa que ela pode ser incluída em um corpo RFC 822 e chegar intacta a seu destino.

Ao receber a mensagem, Bob reverte a codificação base64 e decodifica a chave IDEA utilizando sua chave privada RSA. Empregando essa chave, ele decodifica a mensagem para obter *P1.Z*. Após a descompactação, Bob separa o texto simples do hash criptografado e descriptografa o hash utilizando a chave pública de Alice. Se o hash de texto simples coincidir com seu próprio cálculo MD5, ele saberá que *P* é a mensagem correta e que ela veio de Alice.

Vale a pena observar que o RSA só é usado em duas situações: para criptografar o hash MD5 de 128 bits e para criptografar a chave IDEA de 128 bits. Apesar do RSA ser lento, ele só precisa criptografar 256 bits, e não um grande volume de dados. Além disso, todos os 256 bits de texto simples são excessivamente aleatórios; portanto, Trudy terá muito trabalho para descobrir se uma suposta chave está correta. O trabalho de criptografia é feito pelo IDEA, que é várias ordens de magnitude mais rápido que o RSA. Portanto, o PGP oferece segurança, compactação e assinatura digital de uma forma muito mais eficiente do que o esquema ilustrado na Figura 8.19.

O PGP aceita três tamanhos de chaves RSA. Cabe ao usuário selecionar o mais apropriado. Os tamanhos são:

1. Casual (384 bits): pode ser decifrado com facilidade atualmente.
2. Comercial (512 bits): pode ser decifrado por empresas de informática.
3. Militar (1024 bits): ninguém no planeta consegue decifrar.
4. Alienígena (2.048 bits): não pode ser decifrado por ninguém de outros planetas.

Como o RSA só é usado para efetuar dois pequenos cálculos, todos deveriam usar chaves fortes alienígenas o tempo todo.

O formato de uma mensagem PGP clássica é mostrado na Figura 8.45. A mensagem tem três partes, contendo a chave IDEA, a assinatura e a mensagem, respectivamente. A parte referente à chave contém não só a chave, mas também um identificador de chave, pois os usuários podem ter várias chaves públicas.

[arte: ver original p. 802]

[Dísticos]

[1] Base64

Parte de chave da mensagem

Compactada, criptografada pelo IDEA

Parte da assinatura Parte da mensagem

[2] ID de E_B K_M Cab. de assin. Hora ID de E_A Tipos Hash MD5 Cab. de msg Nome de arq. Hora Mensagem

[3] Criptografada por E_B D_A

[F]Figura 8.45

[FL] Uma mensagem PGP

A parte referente à assinatura contém um cabeçalho, que não nos interessará aqui. O cabeçalho é seguido por um timbre de hora, pelo identificador da chave pública do transmissor — que pode ser usada para descriptografar o hash de assinatura — por algum tipo de informação que identifique os algoritmos utilizados (para permitir que o MD6 e o RSA2 sejam usados quando forem criados) e pelo próprio hash criptografado.

A parte referente à mensagem também contém um cabeçalho, o nome padrão do arquivo a ser usado se o receptor gravar o arquivo no disco, o timbre de hora de criação da mensagem e, por fim, a própria mensagem.

No PGP, o gerenciamento de chaves recebeu muita atenção por ser o tendão de Aquiles de todos os sistemas de segurança. Cada usuário mantém duas estruturas de dados localmente: um anel de chave privada e um anel de chave pública. O **anel de chaves privadas** contém um ou mais pares de chave pública/privada. A razão para aceitar vários pares por usuário é permitir que os usuários alterem suas chaves públicas periodicamente ou quando uma delas for considerada comprometida, sem invalidar as mensagens que estiverem sendo preparadas ou em trânsito. Cada par tem um identificador associado, para que o remetente da mensagem informe ao destinatário qual chave pública foi utilizada para criptografá-la. Os identificadores de mensagem consistem nos 64 bits de baixa ordem da chave pública. Os usuários são responsáveis por evitar conflitos

em seus identificadores de chave pública. As chaves privadas armazenadas no disco são criptografadas com o uso de uma senha especial (arbitrariamente longa) para protegê-las contra ataques sorrateiros.

O **anel de chaves públicas** contém as chaves públicas correspondentes do usuário. Esses anéis são necessários para criptografar as chaves associadas a cada mensagem. Cada entrada do anel de chave pública contém não só a chave pública, mas também seu identificador de 64 bits e uma indicação de até que ponto o usuário confia na chave.

O problema que está sendo resolvido é explicado a seguir. Vamos supor que as chaves públicas sejam mantidas em sistemas BBS. Uma forma de Trudy ler a correspondência secreta de Bob é atacar o BBS e substituir a chave pública de Bob por outra de sua escolha. Quando Alice obtiver a chave que supostamente pertence a Bob, Trudy poderá montar um ataque de brigada de incêndio contra Bob.

Para impedir tais ataques, ou pelo menos minimizar suas conseqüências, Alice precisa saber até que ponto pode confiar no item "chave de Bob" em seu anel de chaves públicas. Se Alice souber que Bob entregou pessoalmente um disquete contendo a chave, ela poderá definir o valor de confiança como o mais alto. Essa é uma abordagem descentralizada e controlada pelo usuário para o gerenciamento de chaves públicas, o que distingue o PGP dos esquemas centralizados de PKI.

No entanto, na prática freqüentemente as pessoas recebem chaves públicas consultando um servidor de chaves confiável. Por essa razão, depois da padronização do X.509, o PGP também passou a admitir esses certificados, bem como o mecanismo tradicional de anel de chaves públicas do PGP. Todas as versões atuais do PGP têm suporte X.509.

[T3] 8.8.2 PEM — Privacy Enhanced Mail

Ao contrário do PGP, criado inicialmente por uma única pessoa, nosso segundo exemplo, o **PEM (Privacy Enhanced Mail)**, desenvolvido no final da década de 1980, é um padrão oficial da Internet descrito em quatro RFCs: da RFC 1421 até a 1424. Grosso modo, o PEM cobre o mesmo território do PGP: privacidade e autenticação para sistemas de correio eletrônico baseados na RFC 822. Todavia, ele também tem algumas diferenças em relação ao PGP em termos de estratégia e tecnologia.

Mensagens enviadas com o uso do PEM primeiro são convertidas em um formato canônico, de modo que todas elas tenham as mesmas convenções sobre espaços em branco (por exemplo, tabulações e espaços de separação). Em seguida, um hash de mensagem é calculado com o MD2 ou o MD5. Depois, a concatenação do hash e da mensagem é criptografada com o DES. Considerando-se a conhecida fraqueza de uma chave de 56 bits, essa escolha é sem dúvida suspeita. A mensagem criptografada pode então ser codificada com o uso da codificação base64 e transmitida ao destinatário.

A exemplo do que acontece com o PGP, cada mensagem é criptografada com uma chave de uso único enviada com a mensagem. A chave pode ser protegida com o RSA ou com o DES triplo, utilizando o EDE.

O gerenciamento de chaves é mais estruturado que no PGP. As chaves têm certificados X.509 emitidos pelas CAs, organizadas em uma hierarquia rígida que começa em uma única raiz. A vantagem desse esquema é que a revogação de certificados é possível, fazendo-se a raiz emitir CRLs periódicas.

O único problema do PEM é que ele jamais foi utilizado, embora já tenha sido lançado há muito tempo. O problema era em grande parte político: quem iria operar a raiz e sob que condições? Não faltavam candidatos, mas muitas pessoas tinham receio de confiar a qualquer empresa a segurança do sistema inteiro. A

candidata mais séria era a RSA Security, Inc., que queria cobrar por certificado emitido. Porém, algumas organizações se recusaram a aceitar essa idéia. Em particular, o governo dos Estados Unidos teve permissão para usar todas as patentes americanas gratuitamente, e empresas fora dos EUA se acostumaram a usar o algoritmo RSA sem pagar por ele (a empresa se esqueceu de patenteá-lo fora dos Estados Unidos). Ninguém ficou entusiasmado com a possibilidade repentina de ter de pagar à RSA Security, Inc., pelo uso de algo que sempre foi utilizado de graça. No final, não foi possível criar nenhuma raiz e o PEM entrou em colapso.

[T3] 8.8.3 S/MIME

O próximo empreendimento da IETF relacionado à segurança de correio eletrônico, foi denominado **S/MIME (Secure/MIME)**, e é descrito nas RFCs 2632 a 2643. Como o PEM, ele oferece autenticação, integridade de dados, sigilo e não repúdio. Ele também é bastante flexível, admitindo uma variedade de algoritmos criptográficos. Considerando-se o nome, não surpreende que o S/MIME se integre bem ao MIME, permitindo que todos os tipos de mensagens sejam protegidos. Foi definida uma grande variedade de novos cabeçalhos MIME, por exemplo, para conter assinaturas digitais.

A IETF definitivamente aprendeu algo com a experiência do PEM. O S/MIME não tem uma estrutura rígida de certificados começando em uma única raiz. Em vez disso, os usuários podem ter várias âncoras de confiança. Desde que a origem de um certificado possa ser acompanhada até alguma âncora de confiança em que o usuário acredite, ele é considerado válido. O S/MIME utiliza os algoritmos e protocolos padrão que examinamos até agora; portanto, não o discutiremos mais aqui. Para ver os detalhes, consulte as RFCs.

[T2] 8.9 Segurança da Web

Acabamos de estudar duas áreas importantes em que a segurança é necessária: comunicações e correio eletrônico. Considere essas a entrada e o aperitivo. Agora, vamos ao prato principal: a segurança da Web. A Web é o lugar em que encontramos a maioria dos intrusos, espionando e fazendo seu trabalho sujo. Nas próximas seções examinaremos alguns problemas e questões relacionadas à segurança da Web.

Grosso modo, a segurança da Web pode ser dividida em três partes. Primeiro, como os objetos e os recursos são nomeados com segurança? Em segundo lugar, como é possível estabelecer conexões seguras e autenticadas? Terceiro, o que acontece quando um Web site envia a um cliente um fragmento de código executável? Depois de estudarmos algumas ameaças, vamos examinar todas essas questões.

[T3] 8.9.1 Ameaças

Quase toda semana, lemos nos jornais notícias sobre problemas de segurança de Web sites. A situação é realmente bastante séria. Vamos examinar alguns exemplos do que já aconteceu. Primeiro, a home page de inúmeras organizações é atacada e substituída por uma nova home page escolhida pelos crackers. (O imprensa popular chama as pessoas que invadem computadores de "hackers", mas muitos programadores reservam esse termo para os ótimos programadores. Preferimos chamar esses invasores de "crackers".) Os sites invadidos incluem Yahoo, o Exército dos Estados Unidos, a CIA, a NASA e o New York Times. Na maioria dos casos, os crackers simplesmente colocavam algum texto engraçado, e os sites eram reparados dentro de algumas horas.

Agora, vamos observar alguns casos muito mais sérios. Numerosos sites foram derrubados por ataques de negação de serviço, nos quais o cracker inunda o site

com tráfego, tornando-o incapaz de responder a consultas legítimas. Com freqüência, o ataque é montado a partir de um grande números de máquinas que o cracker já invadiu (ataques DDoS). Esses ataques são tão comuns que já não geram mais notícias, mas podem custar ao site atacado milhares de dólares em negócios perdidos.

Em 1999, um cracker sueco invadiu o Web site Hotmail da Microsoft e criou um site espelho que permitia a qualquer pessoa digitar o nome de um usuário do Hotmail, e depois ler toda o correio eletrônico atual e arquivado da pessoa.

Em outro caso, um cracker russo de 19 anos chamado Maxim invadiu um Web site de comércio eletrônico e roubou 300.000 números de cartões de crédito. Em seguida, o cracker abordou os proprietários do site e informou que, se recebesse 100.000 dólares, iria postar todos os números de cartões de crédito na Internet. Eles não cederam à chantagem, e o cracker realmente publicou os números dos cartões de crédito, causando muitos danos a muitas vítimas inocentes.

Em um cenário diferente, um aluno da Califórnia de 23 anos enviou por correio eletrônico um comunicado a uma agência de notícias, afirmando que a Emulex Corporation iria anunciar um grande prejuízo trimestral e que o CEO da empresa renunciaria imediatamente. Dentro de poucas horas, as ações caíram 60%, fazendo os acionistas perderem mais de 2 bilhões de dólares. O atacante ganhou 250.000 dólares vendendo suas ações pouco antes de enviar o anúncio. Embora esse evento não represente a invasão de um Web site, é claro que a inserção de um anúncio desse tipo na home page de qualquer grande corporação teria um efeito semelhante.

Poderíamos (infelizmente) continuar como esse assunto por muitas páginas, mas agora devemos examinar algumas questões técnicas relacionadas à segurança da Web. Para obter mais informações sobre problemas de segurança de todos os tipos, consulte (Anderson, 2001; Garfinkel com Spafford, 2002; e Schneier,

2000). A pesquisa na Internet também resultará na apresentação de um grande número de casos específicos.

[T3] 8.9.2 Nomenclatura segura

Vamos iniciar com algo bastante básico: Alice quer visitar o Web site de Bob. Ela digita o URL de Bob em seu navegador e, em alguns segundos, surge uma página da Web. Porém, será a página de Bob? Talvez sim e talvez não. Trudy poderia colocar em prática mais uma vez seus velhos truques. Por exemplo, ela poderia interceptar todos os pacotes enviados por Alice e examiná-los. Quando capturar uma solicitação *GET* de HTTP endereçada ao Web site de Bob, ela mesma pode ir até o Web site de Bob para obter a página, modificá-la como desejar e retornar à Alice a página falsa. Alice nem ficaria sabendo. Pior ainda, Trudy poderia diminuir os preços da loja eletrônica de Bob para tornar suas mercadorias muito atraentes, fazendo Alice enviar seu número de cartão de crédito para "Bob", a fim de adquirir algumas mercadorias.

Uma desvantagem desse clássico ataque de homem em posição intermediária é que Trudy tem de estar em uma posição conveniente para interceptar o tráfego enviado por Alice e forjar seu tráfego de entrada. Na prática, ela tem de bater grampear a linha telefônica de Alice ou de Bob, pois é muito difícil grampear o backbone de fibra óptica. Embora a espionagem ativa certamente seja possível, ela exige um determinado volume de trabalho e, embora seja inteligente, Trudy também é preguiçosa. Além disso, existem maneias mais fáceis de enganar Alice.

[T4] Spoofing de DNS

Por exemplo, suponha que Trudy seja capaz de invadir o sistema DNS, talvez apenas o cache DNS no ISP de Alice, e substitua o endereço IP de Bob (digamos, 36.1.2.3) por seu próprio endereço IP (digamos, 42.9.9.9). Isso leva ao ataque

explicado a seguir. O modo como ele deve funcionar é ilustrado na Figura 8.46(a). Aqui (1) Alice solicita ao DNS o endereço IP de Bob, (2) recebe esse endereço, (3) pergunta a Bob por sua home page e (4) também recebe a home page. Depois de Trudy ter modificado o registro de DNS de Bob para conter seu próprio endereço IP em lugar do endereço de Bob, temos a situação da Figura 8.46(b). Nesse caso, quando Alice procurar o endereço IP de Bob receberá o de Trudy, e então todo o tráfego destinado a Bob irá para Trudy. Agora, Trudy pode montar um ataque de home em posição intermediária sem ter o trabalho de grampear linhas telefônicas. Em vez disso, ela terá de invadir um servidor DNS e alterar um único registro, uma proposta muito mais fácil.

[arte: ver original p. 807]

[Dísticos]

[1] Servidor DNS

Alice 1 2 Servidor da Web de Bob (36.1.2.3)

3

4

1. Dê-me o endereço IP de Bob

2. 36.1.2.3 (endereço IP de Bob)

3. GET index.html

4. Home page de Bob

(a)

[2] Servidor DNS invadido

Alice 1 2 Servidor da Web de Trudy (42.9.9.9)

3

4

1. Dê-me o endereço IP de Bob

2. 42.9.9.9 (endereço IP de Trudy)

3. GET index.html

4. Home page de Bob fraudada por Trudy

(b)

[F]Figura 8.46

[FL] (a) Situação normal. (b) Um ataque baseado na invasão do DNS e na modificação do registro de Bob

Como Trudy poderia enganar o DNS? Na verdade, é relativamente fácil. Em resumo, Trudy pode enganar o servidor DNS no ISP de Alice, enviando-lhe uma consulta do endereço IP de Bob. Infelizmente, como o DNS utiliza o UDP, o servidor DNS não tem nenhum meio real de verificar quem forneceu a resposta. Trudy pode explorar essa propriedade forjando a resposta esperada e, desse modo, injetar um falso endereço IP no cache do servidor DNS. Por simplicidade, vamos supor que o ISP de Alice não tem inicialmente uma entrada para o Web site de Bob, *bob.com*. Se tiver, Trudy pode esperar até ele entrar em timeout e tentar mais tarde (ou usar outros truques).

Trudy inicia o ataque enviando uma solicitação de pesquisa ao ISP de Alice, pedindo o endereço IP de *bob.com*. Tendo em vista que não existe nenhuma entrada correspondente a esse nome DNS, o servidor de cache consulta o servidor de nível superior em busca do domínio *com*, a fim de obter uma entrada. No entanto, Trudy invade o servidor *com* e envia de volta uma resposta falsa que informa: "*bob.com* é 42.9.9.9" mas, na realidade, o endereço IP é o dela. Se sua falsa resposta voltar primeiro ao ISP de Alice, ela será guardada no cache e a resposta real será rejeitada como uma resposta não solicitada a uma consulta que não está mais pendente. Enganar um servidor DNS fazendo-o instalar um falso endereço IP é uma ação chamada **spoofing de DNS**. Um cache que contém um endereço IP intencionalmente falso como esse é chamado **cache envenenado**.

Na realidade, nem tudo é assim tão simples. Primeiro, o ISP de Alice verifica se a resposta contém o endereço IP de origem correto do servidor de nível superior. Porém, como Trudy pode colocar o que quiser nesse campo de endereço IP, ela pode anular com facilidade esse teste, pois os endereços IP dos servidores de nível superior têm de ser públicos.

Em segundo lugar, para permitir que os servidores DNS saibam a resposta correspondente a cada solicitação, todas as solicitações têm um número de seqüência. Para enganar o ISP de Alice, Trudy tem de conhecer seu número de seqüência atual. O modo mais fácil de aprender o número de seqüência atual é Trudy registrar um domínio ela mesma, digamos, *trudy-a-intrusa.com*. Vamos supor que seu endereço IP também seja 42.9.9.9. Ela também cria um servidor DNS para seu novo domínio, *dns.trudy-a-intrusa.com*. Esse servidor utiliza ainda o endereço IP de Trudy, 42.9.9.9, pois Trudy só tem um computador. Agora, ela tem de dar ciência ao ISP de Alice de seu servidor DNS. Isso é fácil. Tudo que ela tem de fazer é pedir ao ISP de Alice que lhe forneça *foobar.trudy-a-intrusa.com*. Isso fará com que o ISP de Alice descubra quem serve ao novo domínio de Trudy, solicitando o endereço ao servidor de endereços *com* de nível superior.

Com *dns.trudy-a-intrusa.com* em segurança no cache do ISP de Alice, o ataque real pode começar. Agora, Trudy consulta o ISP de Alice em busca de *www.trudy-a-intrusa.com*. Naturalmente, o ISP envia ao servidor DNS de Trudy uma consulta solicitando essa página. Essa consulta contém o número de seqüência que Trudy está procurando. Rápida como um coelho, Trudy pede ao ISP de Alice que procure Bob. Ela responde de imediato à sua própria pergunta, enviando ao ISP uma resposta forjada, supostamente do servidor *com* de nível superior, afirmando: "*bob.com* é 42.9.9.9". Essa resposta forjada contém um número de seqüência uma unidade maior que o endereço que ela acabou de receber. Enquanto isso, ela também pode enviar uma segunda falsificação com um número de seqüência

duas unidades mais alto, e talvez mais uma dezena de números de seqüência crescentes. Um deles deverá corresponder. Os restantes serão simplesmente descartados. Quando chegar a resposta forjada de Alice, ele estará no cache; mais tarde quando chegar a resposta real, ele será rejeitado, pois não haverá nenhuma consulta pendente nesse momento.

Agora, quando Alice procurar *bob.com*, será informada de que deve usar 42.9.9.9, o endereço de Trudy. Trudy montou um ataque de homem em posição intermediária bem-sucedido, no conforto de sua própria sala de estar. As várias etapas desse ataque estão ilustradas na Figura 8.47. Para piorar, esse não é o único modo de enganar o DNS. Também existem muitos outros.

[arte: ver original p. 808]

[Dísticos]

[1] Servidor DNS para com

5 7 Cache do ISP de Alice

Trudy

1

2

3

4

6

[2] 1. Procura foobar.trudy-a-intrusa.com

(para forçá-lo no cache do ISP)

2. Procura www.trudy-a-intrusa.com

(para obter o próximo número de seqüência do ISP)

3. Solicita www.trudy-a-intrusa.com

(para transportar o próximo número de seqüência do ISP, n)

4. Rápida como um coelho, procura bob.com

(para forçar o ISP a consultar o servidor com na etapa 5)

5. Consulta legítima solicita bob.com usando seq = n+1

6. Resposta forjada de Trudy: Bob é 42.9.9.9, seq = n+1

7. Resposta real (rejeitada, chegou tarde demais)

[F]Figura 8.47

[FL] Como Trudy engana o ISP de Alice

[T4] DNS seguro

Esse ataque específico pode ser anulado fazendo-se os servidores DNS usarem IDs aleatórias em suas consultas, em vez de simplesmente contarem; porém, parece que toda vez que um furo é vedado, surge outro. O problema real é que o DNS foi projetado em uma época na qual a Internet era um recurso de pesquisa para algumas centenas de universidades e nem Alice, nem Bob, nem Trudy tinham sido convidados para a festa. A segurança não era uma questão importante naquele tempo, mas sim fazer a Internet funcionar. O ambiente mudou de forma radical ao longo dos anos; assim, em 1994, a IETF instalou um grupo de trabalho para tornar o DNS fundamentalmente seguro. Esse projeto é conhecido como **DNSsec (DNS security)**, e seu resultado é apresentado na RFC 2535. Infelizmente, o DNSsec ainda não teve uma distribuição total, e portanto numerosos servidores DNS ainda estão vulneráveis a ataques de spoofing.

Em termos conceituais, o DNSsec é extremamente simples. Ele se baseia na criptografia de chave pública. Cada zona DNS (no sentido da Figura 7.4) tem um par chave pública/chave privada. Todas as informações enviadas por um servidor DNS são assinadas com a chave privada da zona de origem, de forma que o receptor possa verificar sua autenticidade.

O DNSsec oferece três serviços fundamentais:

1. Prova de onde os dados se originaram.

2. Distribuição de chave pública.

3. Autenticação de transação e solicitação.

O principal serviço é o primeiro, que verifica se os dados que estão sendo retornados foram aprovados pelo proprietário da zona. O segundo é útil para armazenar e recuperar chaves públicas com segurança. O terceiro é necessário como proteção contra ataques por reprodução e spoofing. Observe que o sigilo não é um serviço oferecido, pois todas as informações no DNS são consideradas públicas. Tendo em vista que a implantação do DNSsec deverá demorar vários anos, a habilidade de servidores conscientes da segurança para interoperar com servidores que ignoram os aspectos a segurança é algo essencial; isso implica que o protocolo não pode ser alterado. Agora, vamos observar alguns detalhes.

Os registros DNS são agrupados em conjuntos chamados **RRSets (Resource Record Sets)**, com todos os registros que têm o mesmo nome, a mesma classe e o mesmo tipo sendo reunidos em um único conjunto. Por exemplo, um RRSet pode conter vários registros *A*, se um nome DNS for resolvido em um endereço IP primário e um endereço IP secundário. Os RRSets são estendidos com vários tipos novos de registros (descritos a seguir). Cada RRSet passa por um hash criptográfico (por exemplo, com o MD5 ou com o SHA-1). O hash é assinado pela chave privada da zona (por exemplo, usando-se o RSA). A unidade de transmissão para clientes é o RRSet assinado. Ao receber um RRSet assinado, o cliente pode verificar se ele foi assinado pela chave privada da zona de origem. Se a assinatura coincidir, os dados serão aceitos. Tendo em vista que cada RRSet contém sua própria assinatura, os RRSets pode ser armazenados no cache em qualquer lugar, até mesmo em servidores não confiáveis, sem trazer perigo à segurança.

O DNSsec introduz vários tipos novos de registros. Os primeiro deles é o registro *KEY*. Esse registro contém a chave pública de uma zona, um usuário, um host ou

outro protagonista, o algoritmo de criptografia usado na assinatura, o protocolo empregado na transmissão e alguns outros bits. A chave pública é armazenada em estado bruto. Os certificados X.509 não são usados devido a seu tamanho. O campo do algoritmo contém um valor 1 para assinaturas MD5/RSA (a opção preferida) e outros valores para outras combinações. O campo de protocolo pode indicar o uso do IPsec ou de outros protocolos de segurança, se houver.

O segundo dentre os novos tipos de registros é o registro *SIG*. Ele contém o hash assinado de acordo com o algoritmo especificado no registro *KEY*. A assinatura se aplica a todos os registros no RRSet, incluindo quaisquer registros *KEY* presentes, mas excluindo ela própria. Ele também contém os horários em que a assinatura inicia seu período de validade e de vencimento, bem como o nome do signatário e de alguns outros itens.

O projeto do DNSsec é tal que a chave privada de uma zona pode ser mantida off-line. Uma ou duas vezes por dia, o conteúdo do banco de dados de uma zona pode ser transportado manualmente (por exemplo, em CD-ROM) para uma máquina desconectada na qual a chave privada está localizada. Todos os RRSets podem ser assinados nessa máquina e os registros *SIG* assim produzidos podem ser transportados de volta ao servidor primário da zona em CD-ROM. Desse modo, a chave privada pode ser armazenada em um CD-ROM bloqueado de forma segura, exceto quando é inserido na máquina desconectada para assinar os novos RRSets do dia. Depois que o processo de assinatura é concluído, todas as cópias da chave são apagadas da memória, sendo o disco e o CD-ROM devolvidos a um local seguro. Esse procedimento reduz a segurança eletrônica à segurança física, algo que as pessoas entendem como tratar.

Esse método de assinatura prévia de RRSets aumenta bastante a velocidade do processo de responder a consultas, pois nenhuma criptografia tem de ser feita durante a execução. Em compensação, é necessário um grande volume de espaço

de disco para armazenar todas as chaves e assinaturas nos bancos de dados DNS. Alguns registros aumentarão dez vezes em tamanho devido à assinatura.

Quando um processo cliente obtém um RRSet assinado, ele tem de aplicar a chave pública da zona de origem para decifrar o hash, calcular o próprio hash e comparar os dois valores. Se eles concordarem, os dados serão considerados válidos. Porém, esse procedimento faz surgir a seguinte questão: como o cliente obtém a chave pública da zona? Uma alternativa é adquiri-la de um servidor confiável, utilizando uma conexão segura (por exemplo, usando o IPsec).

Porém, na prática, espera-se que os clientes sejam pré-configurados com as chaves públicas de todos os domínios de nível superior. Se agora Alice quiser visitar o Web site de Bob, ela poderá solicitar ao DNS o RRSet de *bob.com*, que conterá seu endereço IP e um registro *KEY* contendo a chave pública de Bob. Esse RRSet será assinado pelo domínio *com* de nível superior, e assim Alice poderá verificar facilmente sua validade. Um exemplo do que esse RRSet pode conter é mostrado na Figura 8.48.

Agora, munida com uma cópia verificada da chave pública de Bob, Alice pode pedir ao servidor DNS de Bob (executado por Bob) o endereço IP de *www.bob.com*. Esse RRSet será assinado pela chave privada de Bob, e assim Alice pode verificar a assinatura de Bob no RRSet que ele retorna. Se Trudy, de alguma maneira, conseguir injetar um falso RRSet em qualquer dos caches, Alice poderá detectar essa falta de autenticidade facilmente, porque o registro *SIG* contido nele será incorreto.

Porém, o DNSsec também fornece um mecanismo criptográfico para vincular uma resposta a uma consulta específica, a fim de impedir o tipo de ataque que Trudy tentou realizar na Figura 8.47. Essa medida (opcional) contra o spoofing adiciona à resposta um hash da mensagem de consulta assinado com a chave privada do autor da resposta. Como Trudy não conhece a chave privada do servidor *com* de

nível superior, ela não pode forjar uma resposta a uma consulta ao ISP de Alice enviada pelo ISP. Sem dúvida, ela pode receber sua resposta de volta primeiro, mas essa resposta será rejeitada devido à assinatura inválida do hash.

[arte: ver original p. 811]

[T]Tabela

Nome de domínio	Validade	Classe	Tipo	Valor
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F73 1 029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

[F]Figura 8.48

[FL] Um exemplo de RRSet para *bob.com*. O registro *KEY* é chave pública de Bob. O registro *SIG* é o hash assinado do servidor *com* de nível superior dos registros *A* e *KEY*, a fim de verificar sua autenticidade

O DNSsec também admite alguns outros tipos de registros. Por exemplo, o registro *CERT* pode ser usado para armazenar certificados (por exemplo, X.509). Esse registro é fornecido, porque algumas pessoas querem transformar o DNS em uma PKI. Resta saber se de fato isso é possível. Interromperemos nossa discussão sobre o DNSsec aqui. Para obter mais detalhes, consulte a RFC 2535.

[T4] Nomes de certificação automática

O DNS seguro não é a única possibilidade para proteger nomes. Uma abordagem completamente diferente é usada no **Secure File System** (Mazières *et al.*, 1999). Nesse projeto, os autores criaram um sistema de arquivos seguro e escalável de âmbito mundial, sem modificar o DNS (padrão) e sem usar certificados ou supor a existência de uma PKI. Nesta seção, mostraremos como suas idéias poderiam ser aplicadas à Web. Conseqüentemente, na descrição a seguir, usaremos a

terminologia da Web em lugar da terminologia de sistemas de arquivos usada no artigo original. Porém, para evitar qualquer confusão possível, embora esse esquema *pudesse* ser aplicado à Web para alcançar alta segurança, ele não está em uso no momento e seriam necessárias mudanças significativas no software para introduzi-lo.

Começaremos supondo que cada servidor da Web tem um par chave pública/chave privada. A essência da idéia é que cada URL contém um hash criptográfico do nome do servidor e da chave pública como parte do URL. Por exemplo, na Figura 8.49 vemos o URL correspondente à foto de Bob. Ele começa com o esquema habitual *http*, seguido pelo nome DNS do servidor (*www.bob.com*). Depois, há um ponto-e-vírgula e um hash de 32 caracteres. No final, tem-se o nome do arquivo, de novo da maneira habitual. Exceto pelo hash, esse é um URL padrão. Com o hash, ele é um **URL de certificação automática.**

[arte: ver original p. 812]

[Dísticos]

[1]

Servidor SHA-1 (Servidor, Chave pública do servidor) Nome do arquivo

http://www.bob.com:2g5hd8bfjkc7mf6hg8dgany23xds4pe6/photos/bob.jpg

[F]Figura 8.49

[FL] Um URL de certificação automática contendo um hash do nome e da chave pública do servidor

A pergunta óbvia é: para que serve o hash? O hash é calculado concatenando-se o nome DNS do servidor com a chave pública do servidor e passando-se o resultado pela função SHA-1, a fim de se obter um hash de 160 bits. Nesse esquema, o hash é representado como uma seqüência de 32 dígitos e letras minúsculas, com exceção das letras "l" e "o" e dos dígitos "1" e "0", para evitar

confusão. Isso nos dá 32 dígitos e letras possíveis. Com 32 caracteres disponíveis, cada um pode codificar um string de 5 bits. Um string de 32 caracteres pode conter o hash SHA-1 de 160 bits. Na realidade, não é necessário usar um hash; a própria chave poderia ser empregada. A vantagem do hash é reduzir o tamanho do nome.

No modo mais simples (embora menos conveniente) de ver a foto de Bob, Alice digita apenas o string da Figura 8.49 em seu navegador. O navegador envia uma mensagem ao Web site de Bob, solicitando sua chave pública. Quando chega a chave pública de Bob, o navegador concatena o nome do servidor e a chave pública, e executa o algoritmo de hash. Se o resultado coincidir com o hash de 32 caracteres no URL seguro, o navegador terá certeza de que possui a chave pública de Bob. Afinal, devido às propriedades do SHA-1, ainda que Trudy intercepte a solicitação e falsifique a resposta, ela não terá como encontrar uma chave pública que forneça o hash esperado. Qualquer interferência de sua parte será então detectada. A chave pública de Bob pode ser armazenada no cache para uso futuro.

Agora, Alice tem de verificar se Bob tem a chave privada correspondente. Ela elabora uma mensagem contendo uma chave de sessão AES proposta, um nonce e um timbre de hora. Em seguida, Alice codifica a mensagem com a chave pública de Bob e a envia a ele. Tendo em vista que somente Bob tem a chave privada correspondente, apenas Bob pode decifrar a mensagem e devolver o nonce codificado com a chave AES. Ao receber o nonce correto codificado com o AES, Alice sabe que está se comunicando com Bob. Além disso, Alice e Bob têm agora uma chave de sessão AES para solicitações *GET* e respostas subseqüentes.

Uma vez Alice tenha a fotografia de Bob (ou qualquer página da Web), ela pode identificá-la com um marcador (bookmark), para não ter de digitar outra vez o URL completo. Além disso, os URLs incorporados a páginas da Web também

podem ser URLs de certificação automática, e portanto podem ser usados com um simples clique sobre eles, mas com a segurança adicional de se saber que a página retornada é a página correta. Outras formas de evitar a digitação inicial dos URLs de certificação automática são: enviá-los por uma conexão segura a um servidor confiável ou incluí-los em certificados X.509 assinados por CAs.

Outra alternativa para obter URLs de certificação automática seria conectar-se a um mecanismo de pesquisa confiável digitando seu URL de certificação automática (na primeira vez) e passando pelo mesmo protocolo descrito anteriormente, o que leva a uma conexão segura e autenticada para o mecanismo de pesquisa confiável. Então, o mecanismo de pesquisa poderia ser consultado, com os resultados aparecendo em uma página assinada repleta de URLs de certificação automática que poderiam ser ativados por cliques, sem a necessidade de digitar longos strings.

Agora, vamos ver como essa abordagem se comporta no caso do spoofing de DNS de Trudy. Se Trudy conseguir envenenar o cache do ISP de Alice, a solicitação de Alice poderá ser entregue a Trudy, e não a Bob. Contudo, agora o protocolo exige que o destinatário de mensagem inicial (isto é, Trudy) retorne uma chave pública que produza o hash correto. Se Trudy retornar sua própria chave pública, Alice descobrirá imediatamente porque o hash do SHA-1 não coincidirá com o URL de certificação automática. Se Trudy retornar a chave pública de Bob, Alice não detectará o ataque, mas codificará sua próxima mensagem, usando a chave de Bob. Trudy receberá a mensagem, mas não terá como decodificá-la para extrair a chave AES e o nonce. De qualquer modo, tudo que o spoofing de DNS poderá fazer será um ataque de negação de serviço.

[T3] 8.9.3 SSL — Secure Sockets Layer

A nomenclatura segura é um bom começo, mas existem vários outros detalhes

sobre segurança da Web. A próxima etapa é gerar conexões seguras. Agora, vamos examinar como as conexões seguras podem ser alcançadas.

Quando a Web repentinamente chegou ao público, ela foi usada no início apenas para distribuir páginas estáticas. Porém, em pouco tempo, algumas empresas tiveram a idéia de usá-la para transações financeiras, como a compra de mercadorias por cartões de crédito, transações bancárias on-line e mercado de capitais eletrônico. Essas aplicações criaram uma demanda por conexões seguras. Em 1995, a Netscape Communications Corp., que então dominava o mercado de fabricantes de navegadores, respondeu introduzindo um pacote de segurança chamado **SSL (Secure Sockets Layer)** para atender a essa demanda. Esse software e seu protocolo agora também são amplamente utilizados pelo Internet Explorer, e portanto vale a pena examiná-los com mais detalhes.

A SSL constrói uma conexão segura entre dois soquetes, incluindo:

1. Negociação de parâmetros entre cliente e servidor.
2. Autenticação mútua de cliente e servidor.
3. Comunicação secreta.
4. Proteção da integridade dos dados.

Já vimos esses itens antes, e portanto não há necessidade de desenvolvê-los.

O posicionamento da SSL na pilha de protocolos habitual é ilustrado na Figura 8.50. Efetivamente, trata-se de uma nova camada colocada entre a camada de aplicação e a camada de transporte, aceitando solicitações do navegador e enviando-as ao TCP para transmissão ao servidor. Depois que a conexão segura é estabelecida, a principal tarefa da SSL é manipular a compactação e a criptografia. Quando o HTTP é usado sobre a SSL, ele se denomina **HTTPS (Secure HTTP)**, embora seja o protocolo HTTP padrão. Às vezes, ele está disponível em uma nova porta (443), em lugar da porta padrão (80). A propósito, a SSL não se limita ao uso apenas com navegadores da Web, mas essa é sua aplicação mais comum.

[arte: ver original p. 814]

[Dísticos]

[1] Aplicação (HTTP)

[2] Segurança (SSL)

[3] Transporte (TCP)

[4] Rede (IP)

[5] Enlace de dados (PPP)

[6] Física (modem, ADSL, TV a cabo)

[F]Figura 8.50

[FL] Camadas (e protocolos) para um usuário doméstico navegando com a SSL

O protocolo SSL passou por várias versões. Descreveremos apenas a versão 3, a versão mais amplamente utilizada. A SSL admite uma variedade de algoritmos e opções distintas. Essas opções incluem a presença ou a ausência de compactação, os algoritmos criptográficos a serem usados e algumas questões relativas a restrições de exportação impostas à criptografia. A última se destina principalmente a assegurar que a criptografia séria será utilizada apenas quando ambas as extremidades da conexão estiverem nos Estados Unidos. Em outros casos, as chaves serão limitadas a 40 bits, que os criptógrafos consideram uma piada. A Netscape foi forçada a colocar essa restrição para obter uma licença de exportação do governo dos Estados Unidos.

A SSL consiste em dois subprotocolos, um para estabelecer uma conexão segura e outro para usá-la. Vamos começar examinando como as conexões seguras são estabelecidas. O subprotocolo de estabelecimento de conexões é mostrado na Figura 8.51. Ele começa com a mensagem 1, quando Alice envia uma solicitação a Bob para estabelecer uma conexão. A solicitação especifica a versão de SSL que Alice tem e suas preferências com relação aos algoritmos de compactação e de

criptografia. Ela também contém um nonce R_A, a ser usado mais tarde.

Agora é a vez de Bob. Na mensagem 2, Bob faz uma escolha entre os diversos algoritmos que Alice pode admitir e envia seu próprio nonce R_B. Em seguida, na mensagem 3, ele envia um certificado contendo sua chave pública. Se esse certificado não for assinado por alguma autoridade conhecida, ele também envia uma cadeia de certificados que pode ser seguida de volta até chegar a uma autoridade original. Todos os navegadores, inclusive o de Alice, são pré-carregados com cerca de 100 chaves públicas; assim, se Bob puder estabelecer uma cadeia ancorada em uma dessas chaves, Alice será capaz de verificar a chave pública de Bob. Nesse momento, Bob pode enviar algumas outras mensagens (como uma solicitação do certificado de chave pública de Alice). Ao terminar, Bob envia a mensagem 4 para dizer a Alice que agora é a vez dela.

Alice responde escolhendo ao acaso uma **chave pré-mestre** de 384 bits e a envia para Bob, codificada com a chave pública de Bob (mensagem 5). A chave de sessão real usada para codificar os dados é derivada da chave pré-mestre combinada com ambos os nonces de modo complexo. Depois que a mensagem 5 é recebida, Alice e Bob são capazes de calcular a chave de sessão. Por essa razão, Alice informa a Bob que ele deve passar para a nova cifra (mensagem 6) e também que ela concluiu o subprotocolo de estabelecimento (mensagem 7). Bob então confirma as menagens de Alice (mensagens 8 e 9).

Porém, embora Alice saiba quem é Bob, Bob não sabe quem é Alice (a menos que Alice tenha uma chave pública e um certificado correspondente a ela, uma situação improvável para um indivíduo). Portanto, a primeira mensagem de Bob pode ser uma solicitação para Alice se conectar usando um nome de login e uma senha estabelecidos anteriormente. No entanto, o protocolo de login está fora do escopo da SSL. Depois que ele é realizado, por quaisquer meios, o transporte de dados pode se iniciar.

[arte: ver original p. 815]

Atenção, produção!

Não foi possível reproduzir os dísticos desta figura. Sugiro que a imagem seja digitalizada integralmente, pois não há nada a traduzir.

[F]Figura 8.51

[FL] Uma versão simplificada do subprotocolo de estabelecimento de conexões da SSL

Como mencionamos antes, a SSL admite vários algoritmos de criptografia. O mais forte usa o DES triplo com três chaves separadas para criptografia e com o SHA-1, a fim de manter a integridade das mensagens. Essa combinação é relativamente lenta, e assim é usada principalmente em aplicações bancárias e outras aplicações nas quais é exigida a mais alta segurança. Para aplicações comuns de comércio eletrônico, é usado o RC4 com uma chave de 128 bits para criptografia, e o MD5 é empregado para autenticação de mensagens. O RC4 utiliza a chave de 128 bits como uma semente e a expande até um número muito maior para uso interno. Em seguida, ele usa esse número interno para gerar um fluxo de chaves. O fluxo de chaves é submetido a uma operação XOR com o texto simples para fornecer uma cifra de fluxo clássica, como vimos na Figura 8.14. As versões de exportação também utilizam o RC4 com chaves de 128 bits, mas 88 dos bits são divulgados ao público para facilitar a violação da cifra.

Para o transporte real, é usado um segundo subprotocolo, como mostra a Figura 8.52. Primeiro, as mensagens do navegador são divididas em unidades de até 16 KB. Se a compactação estiver ativa, cada unidade será então compactada separadamente. Depois disso, uma chave secreta derivada dos dois nonces e da chave pré-mestre é concatenada com o texto compactado, e o resultado passa por um hash com o algoritmo de hash combinado (normalmente, o MD5). Esse

hash é anexado a cada fragmento como o MAC. O fragmento compactado somado ao MAC é então codificado com o algoritmo de criptografia simétrica estabelecido de comum acordo (em geral, por uma operação XOR entre ele e o fluxo de chaves do RC4). Por fim, é anexado um cabeçalho de fragmento e o fragmento é transmitido pela conexão TCP.

[arte: ver original p. 816]

[Dísticos]

[1] Fragmentação

Compactação

MAC adicionado

Criptografia

Cabeçalho adicionado

[2] Mensagem do navegador

[3] Parte 1 Parte 2

[4] Código de autenticação de mensagens

[F]Figura 8.52

[FL] Transmissão de dados com a SSL

No entanto, é importante um alerta. Por ter sido mostrado que o RC4 tem algumas chaves fracas que podem ser facilmente analisadas com o uso da criptografia, a segurança da SSL usando o RC4 é instável. (Fluhrer *et al.*, 2001). Os navegadores que permitem ao usuário escolher o conjunto de cifras devem ser configurados para usar o DES triplo com chaves de 168 bits e o SHA-1 o tempo todo, embora essa combinação seja mais lenta que o RC4 e o MD5.

Outro problema com a SSL é que os protagonistas não podem ter certificados e, mesmo que tenham, eles nem sempre verificam se as chaves que estão sendo usadas correspondem aos certificados.

Em 1996, a Netscape Communications Corp. submeteu a SSL à IETF para padronização. O resultado foi a **TLS (Transport Layer Security)**, descrita na RFC 2246.

As mudanças feitas na SSL foram relativamente pequenas, mas suficientes para a SSL versão 3 e a TLS não conseguirem interoperar. Por exemplo, o modo como a chave de sessão é derivada da chave pré-mestre e dos nonces mudou para tornar a chave mais forte (isto é, mais difícil de ser violada por criptoanálise). A versão da TLS também é conhecida como SSL versão 3.1. As primeiras implementações apareceram em 1999, mas ainda não está claro se a TLS substituirá a SSL na prática, embora ela seja um pouco mais forte. Contudo permanece o problema de chaves RC4 fracas.

[T3] 8.9.4 Segurança do código móvel

A nomenclatura e as conexões são duas áreas de preocupação relacionadas à segurança da Web, mas existem outras. No início, quando as páginas da Web eram apenas arquivos estáticos de HTML, elas não continham código executável. Agora, as páginas freqüentemente contêm pequenos programas, inclusive miniaplicativos Java, controles ActiveX e JavaScripts. Baixar e executar esse **código móvel** é sem dúvida um grande risco de segurança; por essa razão, foram criados vários métodos para minimizá-lo. Agora, veremos rapidamente algumas questões geradas pelo código móvel e algumas abordagens para lidar com ele.

[T4] Segurança de miniaplicativos Java

Os miniaplicativos (applets) Java são pequenos programas em Java, compilados para uma linguagem de máquina orientada para pilhas, chamada **JVM (Java Virtual Machine)**. Eles podem ser colocados em uma página da Web para serem transferidos por download juntamente com a página. Depois que a página é

carregada, os miniaplicativos são inseridos em um interpretador JVM no navegador, como ilustra a Figura 8.53.

[arte: ver original p. 817]

[Dísticos]

[1] Espaço de endereços virtuais

OxFFFFFFFF

[2] Miniaplicativo não confiável

[3] Sandbox

Interpretador

[4] Miniaplicativo confiável

[5] 0 Navegador da Web

[F]Figura 8.53

[FL] Os miniaplicativos podem ser interpretados por um navegador da Web

A vantagem de executar código interpretado sobre código compilado é que cada instrução é examinada pelo interpretador antes de ser executada. Isso dá ao interpretador a oportunidade de verificar se o endereço da instrução é válido. Além disso, as chamadas do sistema também são captadas e interpretadas. A forma como essas chamadas são manipuladas é um assunto relacionado à política de segurança. Por exemplo, se um miniaplicativo for confiável (por exemplo, se ele veio do disco local), suas chamadas do sistema poderão ser executadas sem questionamentos. Porém, se um miniaplicativo não for confiável (por exemplo, se veio da Internet), ele poderá ser encapsulado em uma **sandbox (caixa de areia)** para limitar seu comportamento e bloquear suas tentativas de usar recursos de sistema.

Quando um miniaplicativo tenta usar um recurso do sistema, sua chamada é repassada a um monitor de segurança para aprovação. O monitor examina a

chamada levando em conta a política de segurança local e depois toma a decisão de permiti-la ou rejeitá-la. Desse modo, é possível dar aos miniaplicativos acesso a alguns recursos, mas não a todos. Infelizmente, a realidade é que o modelo de segurança funciona mal e que os bugs que ele contém surgem o tempo todo.

[T4] ActiveX

Os controles ActiveX são programas binários do Pentium que podem ser incorporados às páginas da Web. Quando um deles é encontrado, é realizada uma verificação para saber se ele deve ser executado e, se passar no teste, o programa é executado. Um controle AtiveX não é interpretado ou colocado em uma sandbox de forma alguma; portanto, tem tanto poder como qualquer outro programa do usuário e tem potencial para causar grandes danos. Desse modo, toda a segurança se resume à decisão de executar ou não o controle ActiveX.

O método que a Microsoft escolheu para tomar essa decisão se baseia na idéia de **assinatura de código**. Cada controle ActiveX é acompanhado por uma assinatura digital — um hash do código assinado por seu criador com o uso de criptografia de chave pública. Quando um controle ActiveX se mostra, primeiro o navegador verifica a assinatura para ter certeza de que ele não foi adulterado em trânsito. Se a assinatura estiver correta, o navegador consulta suas tabelas internas para ver se o criador do programa é confiável, ou se existe uma cadeia de confiança que leve de volta até um criador confiável. Se o criador for confiável, o programa será executado; caso contrário, ele não será executado. O sistema da Microsoft para verificar controles ActiveX é chamado **Authenticode**.

É útil comparar as abordagens Java e ActiveX. Com a abordagem Java, não é feita nenhuma tentativa para determinar quem escreveu o miniaplicativo. Em vez disso, um interpretador run-time certifica-se de que ele não executa ações que o proprietário da máquina afirmou que os miniaplicativos não poderiam executar.

Em contraste, no caso da assinatura de código, não é feita nenhuma tentativa de monitorar o comportamento run-time do código móvel. Se veio de uma origem confiável e não foi modificado em trânsito, ele simplesmente é executado. Não há nenhuma tentativa de verificar se o código é malicioso ou não. Se *era intenção* do programador original criar um código que formatasse o disco rígido e depois apagasse a ROM flash para que o computador nunca mais pudesse ser inicializado, isso não importa: se o programador foi certificado como confiável, o código será executado e destruirá o computador (a menos que os controles ActiveX estejam desativados no navegador).

Muitas pessoas têm receio de confiar em uma empresa de software desconhecida. Para demonstrar o problema, um programador em Seattle formou uma empresa de software e conseguiu certificação como origem fidedigna, o que é fácil. em seguida, desenvolveu um controle ActiveX que provocava um desligamento limpo da máquina e distribuiu amplamente seu controle ActiveX. Ele desativou muitas máquinas, mas elas podiam simplesmente ser reinicializadas, e portanto não havia nenhum dano. O programador estava apenas tentando expor o problema ao mundo. A resposta oficial foi revogar o certificado para esse controle ActiveX específico, o que encerrou um breve episódio de forte embaraço, mas o problema subjacente está persiste, à espera de que um programador diabólico o explore (Garfinkel com Spafford, 2002). Tendo em vista que não há como policiar milhares de empresas de software que poderiam desenvolver código móvel, a técnica de assinatura de código é um desastre que pode ocorrer a qualquer momento.

[T4] JavaScript

O JavaScript não tem nenhum modelo de segurança formal, mas tem um longo histórico de implementações inseguras. Cada fornecedor cuida da segurança de

um modo diferente. Por exemplo, a versão 2 do Netscape Navigator usava algo similar ao modelo do Java mas, na versão 4, essa estratégia foi abandonada em favor de um modelo de assinatura de código.

O problema fundamental é que permitir a execução de código estranho em sua máquina é procurar problemas. Do ponto de vista da segurança, é como convidar um assaltante a entrar em sua casa, e depois tentar observá-lo com cuidado para que ele não possa escapar da cozinha para a sala de estar. Se acontecer algo inesperado e você estiver distraído por um momento, poderão surgir conseqüências desastrosas. A tensão aqui é o fato de que o código móvel permite imagens gráficas atraentes e interação rápida, e muitos projetistas de Web sites acham que isso é muito mais importante que a segurança, especialmente quando é a máquina de outra pessoa que está correndo riscos.

[T4] Vírus

Os vírus são outra forma de código móvel. Porém, diferentes dos exemplos anteriores, os vírus sempre chegam sem ser convidados. A diferença entre um vírus e o código móvel comum é que os vírus são desenvolvidos para se reproduzir. Quando um vírus chega, seja através de uma página da Web, em um anexo de correio eletrônico ou de algum outro modo, em geral ele começa infectando programas executáveis no disco. Quando um desses programas é executado, o controle é transferido para o vírus que, em geral, tenta se difundir para outras máquinas, por exemplo, enviando cópias de si mesmo por correio eletrônico para todas as pessoas que têm seus nomes no catálogo de endereços da vítima. Alguns vírus infectam o setor de inicialização do disco rígido; assim, quando a máquina é inicializada, o vírus é executado. Os vírus se tornaram um problema enorme na Internet e causam prejuízos de bilhões de dólares. Não existe nenhuma solução óbvia. Talvez uma nova geração de sistemas

operacionais, inteiramente baseada em microkernels seguros e rígida divisão dos usuários, processos e recursos em compartimentos estanques possa ajudar a resolver o problema.

[T2] 8.10 Questões sociais

A Internet e sua tecnologia de segurança é uma área para a qual convergem as questões sociais, a política pública e a tecnologia, freqüentemente com enormes conseqüências. Apresentaremos a seguir um breve exame de três áreas: privacidade, liberdade de expressão e direito autoral. É desnecessário dizer que aqui trataremos esse assunto de maneira superficial. Para leitura adicional, consulte (Anderson, 2001; Garfinkel com Spafford, 2002; e Schneier, 2000). A Internet também está repleta de material sobre o tema. Basta digitar palavras como "privacidade", "censura" e "direito autoral" em qualquer mecanismo de pesquisa. Além disso, veja alguns links no Web site deste livro.

[T3] 8.10.1 Privacidade

As pessoas têm direito à privacidade? Boa pergunta. A Quarta Emenda da Constituição dos Estados Unidos proíbe o governo de realizar buscas nas casas das pessoas, vasculhar seus documentos e seus bens sem uma boa razão, e continua a restringir as circunstâncias sob as quais devem ser emitidos os mandados de busca. Desse modo, a privacidade é um direito público há mais de 200 anos, pelo menos nos Estados Unidos.

O que mudou na última década foi a facilidade com que os governos podem espionar seus cidadãos e a facilidade com que os cidadãos podem impedir tais atos de espionagem. No Século 18, para o governo realizar buscas nos documentos de um cidadão, ele tinha de enviar um policial a cavalo até a fazenda do cidadão, exigindo a apresentação de certos documentos. Era um procedimento

incômodo. Hoje em dia, as companhias telefônicas e os provedores da Internet fornecem prontamente grampos ao receberem mandados de busca. Isso facilita a vida do policial e ele não corre o risco de cair do cavalo.

A criptografia muda tudo isso. Qualquer pessoa que se dê o trabalho de baixar e instalar o PGP e que utilize uma chave com força de alienígena bem protegida pode ter certeza de que ninguém no universo conhecido poderá ler sua correspondência de correio eletrônico, com ou sem mandado de busca. Os governos entendem bem esse problema e não o apreciam. A privacidade real significa que é muito mais difícil seus agentes espionarem criminosos de todos os tipos, mas também é muito mais difícil espionar jornalistas e adversários políticos. Conseqüentemente, alguns governos restringem ou proíbem o uso ou a exportação de criptografia. Por exemplo, na França, antes de 1999, toda criptografia era proibida, a menos que o governo recebesse as chaves.

A França não estava só. Em abril de 1993, o governo dos Estados Unidos anunciou sua intenção de criar um criptoprocessador em hardware, o **@@@clipper chip**, o padrão para todas as comunicações em rede. Desse modo, diziam, a privacidade dos cidadãos estaria garantida. Ele também mencionava que o chip fornecia ao governo a possibilidade de decodificar todo tráfego por meio de um esquema chamado **custódia de chaves**, que permitia o acesso do governo a todas as chaves. No entanto, ele prometia só espiar quando tivesse um mandado de busca válido. Não é preciso dizer que o resultado foi uma enorme agitação, com os defensores da privacidade denunciando todo o plano e os promotores de justiça elogiando o esquema. Eventualmente, o governo voltou atrás e descartou a idéia.

Há um grande volume de informações sobre privacidade eletrônica disponível no Web site da Electronic Frontier Foudation, em *www.eff.org*.

[T4] Repostadores anônimos

PGP, SSL e outras tecnologias tornam possível duas partes estabelecerem comunicação segura e autenticada, livre de vigilância e interferência de terceiros. Porém, às vezes a privacidade é melhor servida quando *não* há autenticação, tornando a comunicação anônima. O anonimato pode ser interessante em mensagens ponto a ponto, newsgroups ou ambos.

Vamos considerar alguns exemplos. Primeiro, dissidentes políticos que vivem em regimes autoritários muitas vezes desejam se comunicar de forma anônima para escapar da prisão ou de serem assassinados. Em segundo lugar, delitos em muitas organizações corporativas, educacionais, governamentais e outras freqüentemente são expostos por delatores, que muitas vezes preferem permanecer anônimos para evitar represálias. Em terceiro lugar, pessoas com visões sociais, políticas ou religiosas impopulares podem desejar se comunicar umas com as outras por correio eletrônico ou newsgroups, sem se expor. Em quarto lugar, as pessoas podem desejar discutir alcoolismo, doenças mentais, abusos sexuais, pedofilia, ou ainda participar de um newsgroup formado por uma minoria perseguida sem terem de revelar sua identidade. É claro que existem vários outros exemplos.

Vamos considerar um exemplo específico. Na década de 1990, alguns críticos de um grupo religioso não tradicional postaram suas opiniões em um newsgroup da USENET por meio de um **repostador anônimo**. Esse servidor permitiu que os usuários criassem pseudônimos e enviassem mensagens de correio eletrônico ao servidor, que então reencaminhava ou repostava as mensagens usando o pseudônimo; assim, ninguém poderia saber de onde a mensagem veio de fato. Algumas postagens revelavam que as afirmações do grupo religioso eram segredos comerciais e documentos protegidos por direitos autorais. O grupo religioso respondeu informando às autoridades locais que seus segredos

comerciais tinham sido descobertos e seus direitos autorais infringidos, e ambos os crimes tinham origem no servidor que foi localizado. Seguiu-se um processo criminal e o operador do servidor foi compelido a entregar às autoridades as informações de mapeamento que revelavam as verdadeiras identidades das pessoas que tinham feito as postagens. (A propósito, essa não foi a primeira vez que uma religião ficou insatisfeita porque alguém divulgou seus segredos: William Tyndale foi queimado na fogueira 1536 por traduzir a Bíblia para o idioma Inglês).

Um segmento significativo da comunidade da Internet foi ultrajado por essa brecha de confidencialidade. Todos chegaram à conclusão que um repostador anônimo que armazena um mapeamento entre endereços reais de correio eletrônico e pseudônimos (chamado repostador do tipo 1) não vale a pena. Esse caso estimulou diversas pessoas a criarem repostadores anônimos capazes de resistir a ataques de intimação.

Esses novos repostadores, freqüentemente chamados **repostadores cypherpunk**, funcionam da maneira ilustrada a seguir. O usuário produz uma mensagem de correio eletrônico completa, com cabeçalhos RFC 822 (exceto *Form:*, é claro), codifica a mensagem com a chave pública do repostador e a envia ao repostador. Lá, os cabeçalhos RFC 822 externos são extraídos, o conteúdo é decodificado e a mensagem é repostada. O repostador não tem contas e não mantém nenhum log; assim, mesmo que o servidor seja confiscado mais tarde, não conservará nenhum traço de mensagens que tenham passado por ele.

Muitos usuários que desejam anonimato encadeiam suas solicitações por vários repostadores anônimos, como mostra a Figura 8.54. Aqui, Alice deseja enviar a Bob um cartão pelo Dia dos namorados (um cartão realmente anônimo) e para isso ela utiliza três repostadores. Alice redige a mensagem, *M*, e insere um cabeçalho contendo endereço de correio eletrônico de Bob. Em seguida, ela

codifica toda a mensagem com a chave pública do repostador 3, E_3 (indicada na figura pela hachura horizontal). Para isso, ela anexa um cabeçalho com o endereço de correio eletrônico do repostador 3 em texto simples. Essa é a mensagem mostrada entre os repostadores 2 e 3 na figura.

Depois, Alice codifica essa mensagem com a chave pública do repostador 2, E_2 (indicada pela hachura vertical) e acrescenta um cabeçalho de texto simples contendo o endereço de correio eletrônico do repostador 2. Essa mensagem é mostrada entre 1 e 2 na Figura 8.54. Finalmente, ela codifica a mensagem inteira com a chave pública do repostador 1, E_1, e acrescenta um cabeçalho de texto simples com o endereço de correio eletrônico do repostador 1. Essa é a mensagem mostrada à direita de Alice na figura e é a mensagem que ela de fato transmite.

[arte: ver original p. 822]

[Dísticos]

[1] Para 1

Para 2

Para 3

Para Bob

M

[2] Codificada com E_1

[3] Para 2

Para 3

Para Bob

M

[4] Codificada com E_2

[5] Para 3

Para Bob

M

[6] Codificada com E_3

[7] Para Bob

M

[8] Alice 1 2 3 Bob

[9] Repostador anônimo

[F]Figura 8.54

[FL] Como Alice utiliza 3 repostadores para enviar uma mensagem a Bob

Quando a mensagem chega ao repostador 1, o cabeçalho exterior é extraído. O corpo é decodificado e depois enviado por correio eletrônico para o repostador 2. Etapas semelhantes ocorrem nos outros dois repostadores.

Embora seja extremamente difícil para alguém rastrear a mensagem final de volta até Alice, muitos repostadores tomam precauções de segurança adicionais. Por exemplo, eles podem reter as mensagens por um período de tempo aleatório, adicionar ou remover lixo no fim de uma mensagem e ainda reordenar as mensagens, tudo com a finalidade de tornar mais difícil alguém descobrir que saída de mensagem de um repostador corresponde a cada entrada, a fim de frustrar a análise de tráfego. Para ver a descrição de um sistema que representa o estado da arte em correio eletrônico anônimo, consulte (Mazières e Kaashoek, 1998).

O anonimato não se restringe ao correio eletrônico. Também existem serviços que permitem a navegação anônima na Web. O usuário configura seu navegador para usar o @@@anonymizer como um proxy. Daí em diante, todas as solicitações de HTTP vão para o anonymizer, que solicita a página e a devolve. O Web site vê o anonymizer como a origem da solicitação, não o usuário. Tendo em vista que o anonymizer se recusa a manter um log, depois do fato ninguém pode determinar

quem solicitou cada página.

[T3] 8.10.2 Liberdade de expressão

A privacidade se relaciona a indivíduos que desejam restringir o que outras pessoas podem ver sobre eles. Uma segunda questão social importante é a liberdade de expressão e sua oponente, a censura, que está relacionada com o fato de órgãos governamentais desejarem restringir o que os indivíduos podem ler e publicar. Contendo milhões e milhões de páginas, a Web se tornou um paraíso para o censor. Dependendo da natureza e da ideologia do regime, o material proibido pode incluir Web sites que contém quaisquer dos seguintes itens:

1. Material impróprio para crianças ou adolescentes.
2. Ódio que tem como alvos vários grupos étnicos, religiosos, sexuais ou outros.
3. Informações sobre democracia e valores democráticos.
4. Relatos de eventos históricos que contradizem a versão do governo.
5. Manuais de arrombamento, montagem de armas, codificação de mensagens etc.

A justificativa habitual é proibir os sites de má qualidade.

Às vezes, os resultados são inesperados. Por exemplo, algumas bibliotecas públicas instalaram filtros da Web em seus computadores, a fim de torná-los amigáveis para as crianças, bloqueando sites de pornografia. Os filtros vetam os sites contidos em suas listas negras, mas também examinam páginas em busca de palavras obscenas antes de exibi-las. Em Loudoun County, no estado da Virgínia, o filtro bloqueou a busca de informações sobre câncer de mama, porque o filtro encontrou a palavra "mama". O patrono da biblioteca processou o condado de Loudoun. Porém, em Livermore, Califórnia, um pai processou a biblioteca pública por *não* instalar um filtro depois que seu filho de 12 anos

visualizou um site de pornografia. O que é uma biblioteca pode fazer?

Muitas pessoas não percebem que a World Wide Web é de fato uma teia mundial e, portanto, abrange o mundo inteiro. Nem todos os países concordam sobre o que deve ser permito na Web. Por exemplo, em novembro de 2000, um tribunal francês pediu à Yahoo, uma corporação da Califórnia, para impedir que usuários franceses visualizassem leilões de lembranças nazistas no Web site do Yahoo, porque a posse de tal material viola a lei francesa. O Yahoo apelou a um tribunal dos Estados Unidos que o apoiou, mas a questão das leis que se aplicam a cada país está longe de ser definida.

Imagine estas situações: o que aconteceria se algum tribunal em Utah instruísse a França a bloquear Web sites que lidassem com vinhos, porque eles não concordam com as leis mais estritas do estado de Utah sobre bebidas alcóolicas? Suponha que a China exigisse que todos os Web sites cujo tema fosse democracia fossem proibidos por não serem de interesse do Estado. As leis iranianas sobre religião se aplicam à Suécia? A Arábia Saudita pode bloquear Web sites que tratam dos direitos das mulheres? A questão inteira é uma verdadeira caixa de Pandora.

Um comentário relevante de John Gilmore é: "A rede interpreta a censura como algo danoso e procura contorná-la" Para ver uma implementação concreta, considere o **serviço eterno** (Anderson, 1996). Seu objetivo é assegurar que informações publicadas não poderão ser retiradas de circulação ou reescritas, como era comum na União Soviética durante o regime Josef Stalin. Para usar o serviço eterno, o usuário especifica quanto tempo o material é deve ser preservado, paga uma taxa proporcional à sua duração e ao tamanho, e o transfere por upload. Daí em diante, ninguém poderá removê-lo ou editá-lo, nem mesmo o próprio usuário que fez a transferência.

Como tal serviço poderia ser implementado? O modelo mais simples é usar um

sistema não hierárquico, nos qual os documentos armazenados seriam colocados em dezenas de servidores participantes, cada um dos quais receberia uma fração da tarifa, e portanto um incentivo para se unir ao sistema. Os servidores devem estar espalhados por muitas jurisdições legais, a fim de proporcionar a máxima resiliência. Listas de 10 servidores selecionados ao acaso seriam armazenadas com segurança em vários lugares; assim, se alguma delas fosse comprometida, ainda existiriam outras. Uma autoridade disposta a destruir o documento nunca poderia ter certeza de haver encontrado todas as cópias. O sistema também poderia ser elaborado para reparação automática: caso algumas cópias fossem destruídas, os sites restantes tentariam encontrar novos repositórios para substituí-las.

O serviço eterno foi a primeira proposta de um sistema resistente à censura. Desde então, outros esquemas foram propostos e, em alguns casos, implementados. Diversas características novas foram acrescentadas, como criptografia, anonimato e tolerância a falhas. Com freqüência, os arquivos a serem armazenados são divididos em vários fragmentos, com cada fragmento armazenado em muitos servidores. Alguns desses sistemas são o Freenet (Clarke *et al.*, 2002), PASIS (Wylie *et al.*, 2000) e Publius (Waldman *et al.*, 2000). Outro trabalho é relatado em (Serjantov, 2002).

Muitos países estão cada vez mais procurando regulamentar a exportação de itens intangíveis, o que em geral inclui Web sites, software, artigos científicos, correio eletrônico, assistência técnica por telefone e outros. Até mesmo o Reino Unido, que tem uma tradição de séculos de liberdade de expressão, agora está considerando seriamente leis bastante restritivas que, por exemplo, definiriam discussões técnicas entre um professor britânico e seu aluno estrangeiro na University of Cambridge como uma forma de exportação regulamentada que necessita de uma licença governamental (Anderson, 2002). É desnecessário dizer

que tais normas são controvertidas.

[T4] Esteganografia

Em países onde a censura é abundante, os dissidentes com freqüência tentam usar a tecnologia para burlar sua rigidez. A criptografia permite que mensagens secretas sejam enviadas (embora talvez isso não seja legal); porém, se o governo imaginar que Alice é uma pessoa ruim, o mero fato de ela estar se comunicando com Bob pode incluí-lo nessa categoria, pois governos repressivos entendem o conceito de fechamento transitivo, ainda que esses governos não tenham muitos matemáticos entre eles. Os repostadores anônimos podem ajudar mas, se forem proibidos internamente e as mensagens para o exterior exigirem uma licença de exportação do governo, eles não serão muito úteis. No entanto, a Web pode ser de grande auxílio.

Com freqüência, as pessoas que desejam se comunicar secretamente tentem a ocultar o fato de haver qualquer comunicação. A ciência de ocultar mensagens é chamada esteganografia, das palavras gregas que correspondem a "escrita cifrada". Na verdade, os próprios gregos antigos a utilizavam. Heródoto escreveu sobre um general que raspou a cabeça de um mensageiro, tatuou uma mensagem em seu couro cabeludo e deixou o cabelo crescer de novo antes de enviá-lo ao destino. As técnicas modernas são conceitualmente as mesmas, apenas com uma largura de banda mais alta e uma latência mais baixa.

Um caso interessante é o da Figura 8.55(a). Essa fotografia, tirada pelo autor no Quênia, contém três zebras contemplando uma acácia. A Figura 8.55(b) parece ter exatamente as mesmas três zebras e a acácia mas, além disso, ela tem uma atração a mais. A segunda fotografia contém o texto completo de cinco peças de Shakespeare incorporado a ela: *Hamlet*, *Rei Lear*, *Macbeth*, *O Mercador de Veneza* e *Júlio César*. Juntas, essas peças totalizam mais de 700 KB de texto.

Como funciona esse canal esteganográfico? A imagem em cores original tem 1024 × 768 pixels. Cada pixel consiste em três números de 8 bits, cada um representando a intensidade de uma das cores, vermelha, verde e azul, desse pixel. A cor do pixel é formada pela superposição linear das três cores. O método de codificação esteganográfico utiliza o bit de baixa ordem de cada valor de cor RGB como um canal oculto. Desse modo, cada pixel tem espaço para 3 bits de informações secretas, um no valor vermelho, um no valor verde e um no valor azul. Com uma imagem desse tamanho, podem ser armazenados até 1024 × 768 × 3 bits, ou 294.912 bytes de informações secretas.

[arte: ver original p. 825]

[Dísticos]

[1] (a)

[2] (b)

[F]Figura 8.55

[FL] (a) Três zebras e uma árvore. (b) Três zebras, uma árvore e o texto completo de cinco peças de William Shakespeare

O texto completo das cinco peças e uma pequena nota chegam a 734.891 bytes. Primeiro, esse texto foi compactado para cerca de 274 KB com um algoritmo de compactação padrão. A saída compactada foi então criptografada com o uso do IDEA e inserida nos bits de baixa ordem de cada valor de cor. Como podemos ver (ou melhor, como não podemos ver), a existência das informações é completamente invisível. Elas são igualmente invisíveis na versão ampliada e em cores da fotografia. O olho não consegue distinguir com facilidade entre cores de 21 bits e cores de 24 bits.

A visualização das duas imagens em preto e branco com baixa resolução não faz a justiça ao poder dessa técnica. Para lhe dar uma idéia melhor de como a

esteganografia, o autor preparou uma demonstração, incluindo a imagem em cores de alta resolução da Figura 8.55(b) com as cinco peças incorporadas. A demonstração, incluindo as ferramentas para inserir e extrair textos em imagens, pode ser encontrada no Web site deste livro.

Para usar a esteganografia na comunicação não detectada, os dissidentes poderiam criar um Web site repletos de imagens politicamente corretas, como fotografias do Grande Líder, esportes locais, filmes e estrelas de televisão etc. É claro que as figuras estariam recheadas de mensagens esteganográficas. Se as mensagens fossem primeiro compactadas e depois criptografadas, mesmo que alguém suspeitasse de sua presença teria imensa dificuldade para distinguir as mensagens de ruído branco. É lógico que as imagens devem ser novas; copiar uma figura da Internet e alterar alguns bits é um segredo inútil.

As imagens não são de forma alguma o único tipo de suporte para mensagens esteganográficas. Os arquivos de áudio também funcionam bem. Os arquivos de vídeo têm uma enorme largura de banda esteganográfica. Até mesmo o layout e a ordenação de tags em um arquivo HTML podem transportar informações.

Embora tenhamos examinamos a esteganografia no contexto da liberdade de expressão, ela tem vários outros usos. Um uso comum permite que os proprietários de imagens codifiquem mensagens secretas nessas imagens, declarando seus direitos de propriedade. Se tal imagem for roubada e colocada em um Web site, o dono legal poderá revelar a mensagem esteganográfica no tribunal para provar a quem pertence a imagem. Essa técnica é conhecida como **marca d'água**. Ela é descrita em (Piva *et al.*, 2002).

Para obter mais informações sobre a esteganografia, consulte (Artz, 2001; Johnson e Jajoda, 1998; Katzenbeisser e Petitcolas, 2000; e Wayner, 2002).

[T3] 8.10.3 Direitos autorais

A privacidade e a censura são apenas duas áreas nas quais a tecnologia encontra a política pública. Uma terceira área é a dos direitos autorais. Os **direitos autorais** significam a concessão aos criadores de **IP (Intellectual Property)**, incluindo escritores, artistas, compositores, músicos, fotógrafos, cineastas, coreógrafos e outros, do direito exclusivo de explorar sua IP por um certo período de tempo, em geral durante a vida do autor somada a 50 anos ou 75 anos, no caso da propriedade corporativa. Depois de expirar o período de proteção pelos direitos autorais de uma obra, ela passa para o domínio público e qualquer pessoa pode usá-la ou vendê-la como desejar. Por exemplo, o Projeto Gutenberg (*www.promo.net/pg*) colocou milhares de obras de domínio público (por exemplo, obras de Shakespeare, Twain, Dickens) na Web. Em 1998, a pedido de Hollywood, o Congresso dos Estados Unidos estendeu os direitos autorais nos EUA por mais 20 anos. O pessoal do cinema alegava que, sem uma extensão desse período, ninguém criaria mais nada. Em contraste, as patentes duram apenas vinte anos e, mesmo assim, as pessoas não param de apresentar novas invenções.

A discussão sobre os direitos autorais ganhou espaço quando o Napster, um serviço de troca de obras musicais, alcançou 50 milhões de membros. Embora o Napster realmente não copiasse nenhuma música, os tribunais sustentaram que manter um banco de dados central de quem tinha uma cópia de cada música era infração contribuinte, isto é, eles ajudava outras pessoas a infringirem a lei. Apesar de ninguém afirmar que os direitos autorais sejam má idéia (embora muitos reclamem que o processo é muito longo, favorecendo assim as grandes empresas em detrimento do público), a próxima geração de compartilhamento de música já está levantando questões éticas importantes.

Por exemplo, considere uma rede não hierárquica em que pessoas compartilham arquivos legais (música de domínio público, vídeos domésticos, folhetos

religiosos que não representem segredos comerciais etc.) e talvez alguns deles sejam protegidos por direitos autorais. Suponha que todas essas pessoas estejam on-line o tempo todo, por meio de ADSL ou cabo. Cada máquina tem um índice do que está no disco rígido, além de uma lista de outros membros. Alguém que procurar um item específico pode escolher um membro ao acaso e ver se ele tem o item. Caso contrário, a pessoa pode procurar o item em todos os membros da lista desse primeiro membro, e depois em todos os membros das listas desses outros membros e assim por diante. Os computadores são muito bons nesse tipo de trabalho. Tendo encontrado o item, o solicitante simplesmente o copia.

Se o trabalho estiver protegido por direitos autorais, as chances são de que o solicitante esteja infringindo a lei (embora, no caso de transferências internacionais, não esteja claro que lei deve ser aplicada). Entretanto, como classificar o fornecedor? É crime manter no seu no disco rígido uma música pela qual você pagou e baixou legalmente, apenas porque outras pessoas podem encontrá-la? Se você tem uma cabana no campo e um ladrão de IP invade sua cabana levando um notebook e um scanner, copia um livro protegido por direitos autorais e escapa sorrateiramente, você é culpado do crime de deixar de proteger os direitos autorais de outra pessoa?

No entanto, existem mais dificuldades na área de direitos autorais. Há uma grande entre Hollywood e a indústria de informática. Hollywood deseja a proteção rígida de toda a propriedade intelectual, e a indústria de informática não quer ser a polícia a serviço de Hollywood. Em outubro de 1998, o Congresso norte-americano aprovou o **DMCA (Digital Millennium Copyright Act)** que torna crime frustrar qualquer mecanismo de proteção presente em uma obra protegida por direitos autorais ou informar outras pessoas sobre como lográ-lo. Legislação semelhante está surgindo na União Européia. Embora quase ninguém pense que piratas do Extremo Oriente devam ter permissão para duplicar obras protegidas,

muitas pessoas imaginam que o DMCA desloca completamente o equilíbrio entre o interesse do detentor dos direitos autorais e o interesse público.

Vejamos um exemplo prático. Em setembro de 2000, um consórcio da indústria da música encarregado de elaborar um sistema inviolável para venda de obras musicais on-line patrocinou um concurso convidando pessoas a tentarem violar o sistema (exatamente o que deve ser feito no caso de qualquer sistema de segurança novo). Pesquisadores da área de segurança provenientes de várias universidades formaram uma equipe liderada pelo professor Edward Felten de Princeton que aceitou o desafio e conseguiu romper o sistema. Em seguida, eles escreveram um ensaio sobre suas descobertas e o submeteram a uma conferência de segurança do USENIX, onde esse ensaio passou por uma revisão e foi aceito. Antes de apresentar seu trabalho, Felten recebeu uma carta da Recording Industry Association of America que ameaça processar os autores com base no DMCA se eles publicassem o ensaio.

A resposta dos pesquisadores foi abrir um processo pedindo que um tribunal federal decidisse se a publicação de documentos científicos sobre pesquisas na área de segurança ainda era legal. Temendo uma decisão definitiva dos tribunais contra ela, a indústria retirou sua ameaça e o tribunal rejeitou a ação de Felten. Sem dúvida, os fabricantes de discos foram motivados pela fragilidade de sua posição: eles haviam convidado pessoas a tentarem violar seu sistema, e depois ameaçaram processar algumas delas por aceitarem o desafio. Com a ameaça retirada, o ensaio foi publicado (Craver *et al.*, 2001). Uma nova disputa é quase certa.

Uma questão relacionada a essa é a extensão da **doutrina de uso legal**, estabelecida por decisões judiciais em vários países. Essa doutrina afirma que os compradores de uma obra protegida por direitos autorais têm certos direitos limitados de copiar a obra, inclusive o direito de citar partes dela para fins

científicos, usá-la como material didático em escolas ou faculdades e, em alguns casos, criar cópias de reserva para uso pessoal no caso de falha do meio original. Os testes para definir o que constitui uso legal incluem (1) se o uso é comercial, (2) que porcentagem do todo está sendo copiada e (3) o efeito da cópia sobre as vendas da obra. Tendo em vista que o DMCA e leis semelhantes dentro da União Européia proíbem frustrar os esquemas de proteção contra cópia, essas leis também proíbem o uso legal. Na realidade, o DMCA prejudica os direitos históricos dos usuários para dar mais poder aos vendedores de conteúdo. É inevitável uma confrontação.

Outro desenvolvimento na área que reduz a importância até mesmo do DMCA em seu deslocamento do equilíbrio entre os detentores de direitos autorais e os usuários é a **TCPA (Trusted Computing Platform Alliance)** liderada pela Intel e pela Microsoft. A idéia é fazer o chip da CPU e o sistema operacional monitorarem cuidadosamente o comportamento do usuário em diversos aspectos (por exemplo, reprodução de música pirateada) e proibir o comportamento indesejável. O sistema permite até mesmo que os proprietários de conteúdo manipulem remotamente os PCs dos usuários para alterar as regras quando isso for considerado necessário. É desnecessário dizer que as conseqüências sociais desse esquema são imensas. É ótimo que a indústria esteja finalmente prestando atenção à segurança, mas é lamentável que ela esteja inteiramente voltada para impor a lei de direitos autorais, em vez de lidar com vírus, crackers, intrusos e outras questões de segurança com as quais a maioria das pessoas está preocupada.

Em resumo, os legisladores e juristas estarão ocupados tentando equilibrar os interesses econômicos dos proprietários de direitos autorais com o interesse público nos próximos anos. O espaço virtual não é diferente do espaço físico: ele constantemente joga um grupo contra outro, resultando em lutas pelo poder,

litígio e (esperamos) eventualmente algum tipo de resolução, pelo menos até surgir alguma nova tecnologia capaz e romper esse frágil equilíbrio.

[T2] 8.11 Resumo

A criptografia é uma ferramenta que pode ser usada para manter informações confidenciais e garantir sua integridade e autenticidade. Todos os sistemas criptográficos modernos se baseiam no princípio de Kerckhoff de um algoritmo publicamente conhecido e uma chave secreta. Muitos algoritmos criptográficos usam transformações complexas que envolvem substituições e permutações para transformar o texto simples em texto cifrado. Porém, se a criptografia quântica puder se tornar prática, o uso de blocos de uma única vez poderá fornecer sistemas criptográficos verdadeiramente invioláveis.

Os algoritmos criptográficos podem ser divididos em algoritmos de chave simétrica e algoritmos de chave pública. Os algoritmos de chave simétrica desfiguram os bits em uma série de rodadas parametrizados pela chave para transformar o texto simples no texto cifrado. O DES triplo e Rijndael (AES) são os algoritmos de chave simétrica mais populares no momento. Esses algoritmos podem ser usados em modo de livro de código eletrônico, em modo de encadeamento de blocos de cifras, modo de cifra de fluxo, em modo de contador e outros.

Nos algoritmos de chave pública, são usadas chaves diferentes para codificação e decodificação, e a chave de decodificação não pode ser derivada a partir da chave de codificação. Essas propriedades tornam possível divulgar a chave pública. O principal algoritmo de chave pública é o RSA, cuja força deriva da grande dificuldade de fatorar números extensos.

Documentos legais, comerciais e outros precisam ser assinados. Conseqüentemente, foram criados vários esquemas de assinaturas digitais,

empregando algoritmos de chave simétrica e algoritmos de chave pública. Em geral, as mensagens que devem ser assinadas são submetidas a um hash com a utilização de algoritmos como MD5 ou SHA-1, e então o hash é assinado em lugar das mensagens originais.

O gerenciamento de chaves públicas pode ser implementado com o emprego de certificados, documentos que vinculam um protagonista a uma chave pública. Os certificados são assinados por uma autoridade confiável ou por alguém aprovado (recursivamente) por uma autoridade confiável. A raiz da cadeia tem de ser obtida com antecedência, mas os navegadores em geral têm muitos certificados de raiz embutidos.

Essas ferramentas criptográficas podem ser usadas para proteger o tráfego de rede. O IPsec opera na camada de rede, codificando fluxos de pacotes de host para host. Os firewalls podem efetuar a triagem do tráfego que entra ou sai de uma organização, muitas vezes com base no protocolo e na porta utilizados. As redes privadas virtuais podem simular uma antiga rede de linha dedicada para oferecer certas propriedades de segurança interessantes. Por fim, as redes sem fios precisam de uma boa segurança, e o WEP do padrão 802.11 não a oferece, embora o 802.11i deva melhorar consideravelmente a situação.

Quando duas partes estabelecem uma sessão, elas têm de autenticar uma à outra e, se necessário, estabelecer uma chave de sessão compartilhada. Existem diversos protocolos de autenticação, incluindo alguns que usam uma terceira parte confiável, Diffie-Hellman, Kerberos e criptografia de chave pública.

A segurança de correio eletrônico pode ser alcançada por uma combinação das técnicas que estudamos neste capítulo. Por exemplo, o PGP compacta as mensagens, depois as codifica usando o IDEA. Ele envia a chave do IDEA codificada com a chave pública do receptor. Além disso, ele também efetua o hash da mensagem e envia o hash assinado para confirmar a integridade da

mensagem.

A segurança da Web também é um tópico importante, começando com a nomenclatura segura. O DNSsec oferece um modo de evitar o spoofing de DNS, bem como realizar a certificação automática de nomes. A maioria dos sites de e-commerce na Web utiliza a SSL para estabelecer sessões autenticadas e seguras entre o cliente e o servidor. São usadas várias técnicas para lidar com código móvel, em especial sandboxes e assinatura de código.

A Internet levanta muitas questões em que a tecnologia interage fortemente com a política pública. Algumas áreas relevantes incluem privacidade, liberdade de expressão e direitos autorais.

[T2] Problemas

1. Resolva a cifra monoalfabética a seguir. O texto simples, formado apenas por letras, é um trecho de um conhecido poema de Lewis Carroll.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foija mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Resolva a seguinte cifra de transposição de colunas. O texto simples foi extraído de um livro sobre computadores; portanto, "computer" é uma palavra muito provável. O texto simples é formado apenas por letras (sem espaços). O texto cifrado está dividido em blocos de cinco caracteres para proporcionar melhor legibilidade.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lbolr otoex

3. Encontre um bloco de 77 bits que gere o texto "Donald Duck" a partir do texto cifrado da Figura 8.4.

4. A criptografia quântica exige uma pistola de fótons que possa, por demanda, disparar um único fóton transportando 1 bit. Neste problema, calcule quantos fótons um bit transporta em um enlace de fibra de 100 Gbps. Suponha que o comprimento de um fóton seja igual a seu comprimento de onda que, para fins desse problema, é 1 mícron. A velocidade da luz na fibra é 20 cm/ns.

5. Se Trudy capturar e regenerar fótons quando a criptografia quântica estive em uso, ela obterá alguns fótons errados e provocará o surgimento de erros no bloco de uma só vez de Bob. Em média, que fração dos bits do bloco de uma só vez de Bob estarão errados?

6. Um princípio criptográfico fundamental estabelece que todas as mensagens devem ter redundância. Porém, também sabemos que a redundância ajuda um intruso a saber se uma chave hipotética está correta. Considere duas formas de redundância. Primeiro, os *n* bits iniciais do texto simples contêm um padrão conhecido. Segundo, os *n* bits finais da mensagem contêm um hash sobre a mensagem. Do ponto de vista da segurança, essas duas alternativas são equivalentes? Comente sua resposta.

7. Na Figura 8.6, as caixas P e S se alternam. Apesar da aparência esteticamente agradável dessa organização, não seria mais seguro primeiro ter todas as caixas P e depois todas as caixas S?

8. Projete um ataque ao DES baseado no conhecimento de que o texto simples é formado exclusivamente por letras ASCII em caixa alta, além de espaço, vírgula, ponto, ponto-e-vírgula, retorno de cursor e avanço de linha. Nada é conhecido sobre os bits de paridade do texto simples.

9. No texto, calculamos que uma máquina de análise de cifras com um bilhão de processadores capaz de analisar uma chave em 1 picossegundo demoraria apenas

10^{10} anos para romper a versão de 128 bits do AES. Porém, as máquinas atuais podem ter 1024 processadores e demorar 1 ms para analisar uma chave; assim, precisamos de um aumento de 10^{15} vezes no desempenho apenas para conseguir igualar a máquina de análise de cifras do AES. Se a lei de Moore (a potência de computação duplica a cada 18 meses) continuar a ser válida, quantos anos serão necessários apenas para construir a máquina?

10. O AES admite uma chave de 256 bits. Quantas chaves tem o AES-256? Veja se é possível encontrar algum número em física, química ou astronomia com aproximadamente o mesmo tamanho. Use a Internet para ajudá-lo a procurar números grandes. Tire uma conclusão de sua pesquisa.

11. Suponha que uma mensagem tenha sido criptografada com a utilização do DES no modo de encadeamento de blocos de texto cifrado. Um bit do texto cifrado no bloco C_i é acidentalmente transformado de 0 para 1 durante a transmissão. Quanto do texto simples será adulterado em decorrência disso?

12. Agora, considere mais uma vez o encadeamento de blocos de texto cifrado. Em vez de um único bit 0 ser transformado em um bit 1, um bit 0 extra é inserido no fluxo de texto cifrado depois do bloco C_i. Que proporção do texto simples será adulterado em decorrência disso?

13. Compare o encadeamento de blocos de cifras com o modo de feedback de cifra no que se refere ao número de operações necessárias para a transmissão de um arquivo muito grande. Qual dos dois é o mais eficiente e em que proporção?

14. Usando o sistema de criptografia de chave pública RSA, com $a = 1$, $b = 2$ etc.,

(a) Se $p = 7$ e $q = 11$, cite cinco valores válidos para d.

(b) Se $p = 13$, $q = 31$ e $d = 7$, encontre e.

(c) Usando $p = 5$, $q = 11$ e $d = 27$, encontre e e criptografe "abcdefghij".

15. Suponha que uma usuária, Maria, descubra que sua chave privada RSA ($d1$, $n1$) é igual à chave pública RSA ($e2$, $n2$) de outra usuária, Frances. Em outras

palavras, $d1 = e2$ e $n1 = n2$. Maria deve considerar a hipótese de trocar suas chaves pública e privada? Explique sua resposta.

16. Considere o uso do modo de contador, como mostra a Figura 8.15, mas com $IV = 0$. O uso de 0 ameaça a segurança da cifra em geral?

17. O protocolo de assinatura da Figura 8.18 tem a seguinte falha: se o computador de Bob travar, ele poderá perder o conteúdo de sua memória RAM. Quais problemas isso pode causar e o que pode ser feito para evitá-los?

18. Na Figura 8.20, vemos como Alice pode enviar a Bob uma mensagem assinada. Se Trudy substituir *P*, Bob poderá descobrir. No entanto, o que acontecerá se Trudy substituir ao mesmo tempo *P* e a assinatura?

19. As assinaturas digitais têm uma deficiência potencial devido a usuários preguiçosos. Em transações de comércio eletrônico, um contrato poderia ser interrompido e o usuário poderia ser solicitado a assinar seu hash SHA-1. Se o usuário não verificar realmente que o contrato e o hash correspondem, ele poderá assinar inadvertidamente um contrato diferente. Suponha que a máfia tente explorar essa fraqueza para ganhar algum dinheiro. Os mafiosos configuram um Web site de pagamento (por exemplo, pornografia, jogo etc.) e pedem aos novos clientes um número de cartão de crédito. Em seguida, eles enviam um contrato ao cliente confirmando que este deseja usar seus serviços e que pagará com o cartão de crédito; depois, solicitam que o cliente assine o contrato, sabendo que a maioria dos clientes simplesmente assinará sem verificar se o contrato e o hash coincidem. Mostre como a máfia pode comprar diamantes pela Internet de um joalheiro legítimo e debitá-los de clientes desatentos.

20. Uma turma de matemática tem 20 alunos. Qual é a probabilidade de pelo menos dois alunos fazerem aniversário no mesmo dia? Suponha que ninguém tenha nascido no dia 29 de fevereiro; então, há 365 dias de aniversário possíveis.

21. Depois de Ellen ter confessado a Marilyn que a enganou no episódio da

indicação de Tom, Marilyn resolveu evitar esse problema ditando o conteúdo de futuras mensagens a um gravador e fazendo sua nova secretária simplesmente digitá-las. Marilyn planejava examinar as mensagens em seu terminal depois de serem digitadas, para ter certeza de que continham suas palavras exatas. A nova secretária ainda pode usar o ataque de aniversário para falsificar uma mensagem? De que maneira? *Dica*: Ela pode fazê-lo.

22. Considere a tentativa malsucedida de Alice de conseguir a chave pública de Bob na Figura 8.23. Suponha que Bob e Alice já compartilhem uma chave secreta, mas Alice ainda quer a chave pública de Bob. Agora existe um modo de obtê-la em segurança? Em caso afirmativo, como?

23. Alice quer se comunicar com Bob, usando a criptografia de chave pública. Ela estabelece uma conexão com alguém que espera que seja Bob. Alice pede a ele sua chave pública e ele a envia em texto simples, juntamente com um certificado X.509 assinado pela CA raiz. Alice já tem a chave pública da CA raiz. Que etapas Alice deve executar para verificar se ela está se comunicando com Bob? Suponha que Bob não se importe em saber com quem está se comunicando (por exemplo, Bob é alguma espécie de serviço público).

24. Suponha que um sistema utilize a PKI baseada em uma hierarquia estruturada em árvore de CAs. Alice quer se comunicar com Bob e recebe um certificado de Bob assinado por uma CA *X*, depois de estabelecer um canal de comunicação com Bob. Suponha que Alice nunca tenha ouvido falar em *X*. Que etapas Alice deve executar para confirmar que está se comunicando com Bob?

25. O IPsec usando a AH pode ser empregando em modo de transporte quando uma das máquinas está atrás de uma caixa NAT? Explique sua resposta.

26. Apresente uma vantagem de HMACs sobre o uso do RSA para assinar hashes SHA-1.

27. Apresente uma razão que justifique o fato de um firewall poder ser

configurado de modo a inspecionar o tráfego de entrada. Apresente uma razão que justifique o fato de um firewall poder ser configurado de modo a inspecionar o tráfego de saída. Você acredita que as inspeções têm probabilidade de sucesso?

28. O formato de pacotes WEP é mostrado na Figura 8.31. Suponha que o total de verificação tenha 32 bits, calculado por uma operação XOR de todas as palavras de 32 bits da carga útil. Suponha também que os problemas com o RC4 sejam corrigidos substituindo-se esse algoritmo por uma cifra de fluxo sem deficiências, e que os IVs sejam estendidos até 128 bits. Existe algum meio para um intruso espionar ou interferir com o tráfego sem ser detectado?

29. Suponha que uma organização utilize uma VPN para se conectar em segurança a seus sites pela Internet. Existe a necessidade do usuário Jim dessa organização usar a criptografia ou qualquer outro mecanismo de segurança para se comunicar com a usuária Mary da mesma organização?

30. Faça uma pequena alteração em uma mensagem no protocolo da Figura 8.34, de modo a torná-la resistente ao ataque por reflexão. Explique por que sua mudança funciona.

31. A troca de chaves de Diffie-Hellman está sendo usada para estabelecer uma chave secreta entre Alice e Bob. Alice envia a Bob a mensagem (719, 3, 191). Bob responde com (543). O número secreto de Alice, x, é 16. Qual é a chave secreta?

32. Mesmo que Alice e Bob nunca se encontrem, não compartilhem nenhum segredo e não tenham nenhum certificado, eles podem estabelecer uma chave secreta compartilhada usando o algoritmo de Diffie-Hellman. Explique por que é muito difícil se defender contra um ataque de homem em posição intermediária.

33. No protocolo da Figura 8.39, por que A é enviado em texto simples junto com a chave de sessão criptografada?

34. No protocolo da Figura 8.39, mostramos que iniciar cada mensagem de texto simples com 32 bits zero é um risco de segurança. Suponha que cada mensagem

comece com um número aleatório por usuário, na verdade uma segunda chave secreta conhecida somente por seus usuários e pelo KDC. Isso elimina o problema do ataque por texto simples conhecido? Por quê?

35. No protocolo de Needham-Schroeder, Alice gera dois desafios, R_A e R_{A2}. Isso parece exagero. Apenas um não seria suficiente?

36. Suponha que uma organização utilize o Kerberos para autenticação. Em termos de segurança e disponibilidade de serviço, qual será o efeito se AS ou TGS for desativado?

37. No protocolo de autenticação por chave pública da Figura 8.43, na mensagem 7, R_B é criptografado com K_S. Essa criptografia é necessária, ou teria sido melhor enviar a mensagem de volta em texto simples? Explique sua resposta.

38. Terminais de pontos de venda que utilizam cartões com tarja magnética e códigos PIN têm uma falha fatal: um comerciante inescrupuloso pode modificar sua leitora de cartões para armazenar todas as informações do cartão, assim como o código PIN, a fim de informar outras transações (falsas) no futuro. A próxima geração de terminais de pontos de venda utilizará cartões com uma CPU completa, teclado e um pequenino visor. Imagine um protocolo para esse sistema que comerciantes inescrupulosos não consigam burlar.

39. Cite *duas* razões para o PGP compactar mensagens.

40. Supondo que todos na Internet usassem o PGP, uma mensagem PGP poderia ser enviada a um endereço arbitrário da Internet e ser decodificada corretamente por todos os envolvidos? Comente sua resposta.

41. O ataque mostrado na Figura 8.47 omite uma etapa. A etapa não é necessária para o spoofing funcionar, mas incluí-la poderia reduzir a suspeita potencial depois do fato. Qual é a etapa omitida?

42. Foi apresentada uma proposta para frustrar o spoofing de DNS que utiliza o prognóstico de ID, fazendo-se o servidor inserir uma ID aleatória em vez e usar

um contador. Discuta os aspectos de segurança dessa abordagem.

43. O protocolo de transporte de dados de SSL envolve dois nonces, bem como uma chave de pré-mestre. Que valor, se for o caso, tem o uso dos nonces?

44. A imagem da Figura 8.55(b) contém o texto ASCII de cinco peças de Shakespeare. Seria possível ocultar música em vez de texto entre as zebras? Em caso afirmativo, como isso seria feito e que quantidade de música você ocultaria na imagem? Em caso negativo, por que não?

45. Alice era uma usuária pesada de um repostador anônimo do tipo 1. Ela postava muitas mensagens em seu newsgroup favorito, *alt.fanclub.alice*, e todo mundo sabia que as mensagens eram todas de Alice, porque tinham o mesmo pseudônimo. Supondo-se que o repostador funcionasse corretamente, Trudy não conseguiria se fazer passar por Alice. Depois que os repostadores do tipo 1 foram desativados, Alice trocou para um repostador cypherpunk e iniciou um novo thread de mensagens em seu newsgroup. Elabore um meio de impedir que Trudy possa postar novas mensagens para o newsgroup, fazendo-se passar por Alice.

46. Procure na Internet algum caso interessante envolvendo privacidade e escreva um relatório de uma página sobre o tema.

47. Procure na Internet algum caso jurídico envolvendo direitos autorais *versus* uso legal e escreva um relatório de uma página resumindo os resultados da sua pesquisa.

48. Escreva um programa que codifique sua entrada por meio de uma operação XOR entre ela e um fluxo de chaves. Descubra ou desenvolva o melhor gerador de números aleatórios que puder para gerar o fluxo de chaves. O programa deve atuar como um filtro, recebendo texto simples na entrada padrão e gerando texto cifrado na saída padrão (e vice-versa). O programa deve receber um parâmetro, a chave que produz a semente do gerador de números aleatórios.

49. Escreva um procedimento que calcule o hash SHA-1 de um bloco de dados. O procedimento deve ter dois parâmetros: um ponteiro para o buffer de entrada e um ponteiro para um buffer de saída de 20 bytes. Para ver a especificação exata do SHA-1, procure na Internet FIPS 180-1, a especificação completa.

[TA1]Capítulo

[TA2]9

[T1]Sugestões de leitura e bibliografia

Acabamos de estudar as redes de computadores, mas isso é apenas o começo. Muitos assuntos interessantes não foram tratados com os detalhes merecidos, e outros foram completamente omitidos por falta de espaço. Neste capítulo, apresentaremos algumas sugestões de leitura adicional e uma bibliografia para os usuários que desejarem continuar seus estudos sobre redes de computadores.

[T2] 9.1 Sugestões de leitura adicional

Existe uma extensa literatura sobre todos os aspectos das redes de computadores. Três periódicos que publicam com freqüência artigos sobre essa são *IEEE Transactions on Communications*, *IEEE Journal on Selected Areas in Communications* e *Computer Communication Review*. Muitos outros periódicos também publicam artigos ocasionais sobre o assunto.

O IEEE também publica três revistas – *IEEE Internet Computing*, *IEEE Network Magazine* e *IEEE Communications Magazine* — que contêm pesquisas, tutoriais, e estudos de casos sobre interligação de redes. As duas primeiras enfatizam a arquitetura, os padrões e o software, e a última trata principalmente da tecnologia de comunicações (fibra óptica, satélites e assim por diante).

Além disso, existem várias conferências anuais ou semestrais que atraem numerosos documentos sobre redes e sistemas distribuídos, em particular *SIGCOMM Annual Conference*, *The International Conference on Distributed Computer Systems* e *The Symposium on Operating Systems Principles*.

Vamos relacionar a seguir algumas sugestões para leitura adicional, organizadas de acordo com os capítulos do livro. Em sua maioria esses documentos são

tutoriais ou pesquisas sobre o assunto. Alguns são capítulos de livros didáticos.

[T3] 9.1.1 Introdução e trabalhos genéricos

Bi *et al.*, "Wireless Mobile Communications at the Start of the 21st Century"

Um novo século, uma nova tecnologia. Parece uma boa idéia. Depois de um histórico das redes sem fios, os principais tópicos são abordados aqui, inclusive padrões, aplicações, Internet e tecnologias.

Comer, *The Internet Book*

Se estiver procurando uma introdução de fácil leitura ao estudo da Internet, leia esse livro. Comer descreve a história, o crescimento, a tecnologia, os protocolos e os serviços da Internet em termos simples para os iniciantes, mas o livro tem um volume tão grande de assuntos que também irá interessar aos leitores mais técnicos.

Garber, "Will 3G Really Be the Next Big Wireless Technology?"

Os telefones celulares de terceira geração devem combinar voz e dados e oferecer taxas de dados de até 2 Mbps. Eles demoraram a se firmar no mercado. As promessas, as armadilhas, a tecnologia, a política e a economia da utilização de comunicações de banda larga sem fios são os assuntos abordados nesse artigo de fácil leitura.

IEEE Internet Computing, janeiro/fevereiro de 2000

O primeiro número da revista *IEEE Internet Computing* no novo milênio fez exatamente o que se esperava: pediu que as pessoas que ajudaram a criar a Internet no milênio anterior especulassem sobre a direção que ela iria seguir no próximo milênio. Os especialistas são Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy e outros. Para obter melhores resultados, aguarde 500 anos e *depois* leia suas previsões.

Kipnis, "Beating the System: Abuses of the Standards Adoption Process"

Os comitês de padrões procuram ser justos e neutros em relação aos fornecedores quando realizam seu trabalho mas, infelizmente, há empresas que tentam abusar do sistema. Por exemplo, muitas vezes uma empresa ajuda a desenvolver um padrão, e depois que ele é aprovado, a empresa anuncia que o padrão se baseia em uma patente de sua propriedade e que irá licenciá-lo para as empresas que desejar e não para outras empresas, e ainda que esse licenciamento será feito com preços que ela própria determinar. Se quiser examinar o lado escuro da padronização, você encontrará nesse artigo um excelente ponto de partida.

Kyas e Crawford, *ATM Networks*

O ATM já foi considerado o protocolo de redes do futuro, e ainda é importante no sistema de telefonia. Esse livro é um guia atualizado sobre o status atual do ATM, com informações detalhadas sobre protocolos ATM e como eles podem se integrar a redes baseadas no IP.

Kwok, "A Vision for Residential Broadband Service"

Se quiser saber mais sobre o que a Microsoft pensava sobre os serviços de vídeo sob demanda em 1995, leia esse artigo. Cinco anos mais tarde, essa visão se tornou completamente obsoleta. O valor do artigo é demonstrar que até mesmo pessoas com um elevado grau de conhecimento e bem motivadas não podem fazer previsões sequer para os próximos cinco anos com algum grau de exatidão. Ele deve ser uma lição para todos nós.

Naughton, *A Brief History of the Future*

Afinal, quem criou a Internet? Muitas pessoas reclamam o crédito. Por questão de justiça, devemos admitir que muitas pessoas fizeram parte desse processo de criação, de várias maneiras. Essa história da Internet conta como tudo aconteceu com um estilo engenhoso e encantador, repleto de anedotas. Por exemplo, o livro

relata como a AT&T reiterava sua convicção de que a comunicação digital não tinha nenhum futuro.

Perkins, "Mobile Networking in the Internet"

Para conhecer uma boa avaliação de cada camada de protocolo de redes móveis, leia esse artigo. São estudadas as camadas, desde a física até a de transporte, além do hardware intermediário, da segurança e da interligação de redes *ad hoc*.

Teger e Waks, "End-User Perspectives on Home Networking"

As redes domésticas não são como as redes corporativas. As aplicações são diferentes (utilizam mais intensamente a multimídia), o equipamento vem de uma variedade mais ampla de fornecedores, e os usuários têm pouco treinamento técnico e nenhuma paciência em relação a quaisquer falhas. Para obter mais informações, leia esse artigo.

Varshney e Vetter, "Emerging Mobile and Wireless Networks"

Outra introdução à comunicação sem fio. O artigo focaliza as LANs sem fios, loops locais sem fios e satélites, bem como alguns exemplos de software e aplicações.

Wetteroth, *OSI Reference Model for Telecommunications*

Embora os protocolos OSI propriamente ditos não sejam mais usados, o modelo de sete camadas se tornou bastante conhecido. Além de explicar mais detalhes sobre o OSI, esse livro aplica o modelo às redes de telecomunicações (em vez de usá-lo em redes de computadores), mostrando onde a telefonia comum e outros protocolos de voz se ajustam à pilha de interligação de redes.

[T3] 9.1.2 A camada física

Abramson, "Internet Access Using VSATs"

Pequenas estações terrestres estão se tornando mais populares, tanto para telefonia rural quanto para acesso corporativo à Internet em países

desenvolvidos. Porém, a natureza do tráfego para esses dois casos difere drasticamente, e assim são necessários protocolos distintos para lidar com os dois casos. Nesse artigo, o criador do sistema ALOHA descreve numerosos métodos de alocação de canais que podem ser usados em sistemas VSAT.

Alkhatib *et al.*, "Wireless Data Networks: Reaching the Extra Mile"

Para uma introdução rápida aos termos e às tecnologias de redes sem fios, incluindo o espectro de propagação, esse tutorial é um excelente ponto de partida.

Azzam e Ransom, *Broadband Access Technologies*

O sistema de telefonia, a fibra óptica, ADSL, redes a cabo, satélites e até mesmo linhas de transmissão de energia são assuntos focalizados nesse livro como tecnologias de acesso a redes. Outros tópicos incluem redes domésticas, serviços, desempenho de rede e padrões. O livro termina com análises das principais empresas no ramo de telecomunicações e redes; porém, com a velocidade das mudanças na indústria, esse capítulo talvez tenha uma validade mais curta que os capítulos sobre a tecnologia.

Bellamy, *Digital Telephony*

Tudo que você sempre quis saber sobre o sistema de telefonia e muito mais encontra-se nesse excelente livro. São particularmente interessantes os capítulos sobre transmissão e multiplexação, comutação digital, fibra óptica, telefonia celular e DSL.

Berezdivin *et al.*, "Next-Generation Wireless Communications Concepts and Technologies"

Essa gente está sempre à frente das outras pessoas. A "próxima geração" do título se refere às redes sem fios de quarta geração. Espera-se que essas redes forneçam serviço IP em todos os lugares, com conectividade uniforme para a Internet, incluindo alta largura de banda e excelente qualidade de serviço. Esses

objetivos deverão ser alcançados por uso de uma utilização inteligente do espectro, do gerenciamento dinâmico de recursos e de serviços adaptáveis. Tudo isso parece visionário no momento, mas os telefones celulares também pareciam uma idéia visionária em 1995.

Dutta-Roy, "An Overview of Cable Modem Technology and Market Perspectives"
A TV a cabo (ou por assinatura) deixou de ser algo simples para se transformar em um sistema complexo de distribuição de TV, Internet e telefonia. Essas mudanças afetaram consideravelmente a infra-estrutura de cabos. Para conhecer a instalação de cabos, os padrões e o marketing dessa tecnologia, com ênfase em DOCSIS, vale a pena ler esse artigo.

Farserotu e Prasad, "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends"
Uma variedade de satélites de comunicação de dados está no céu ou na prancheta de desenho, entre eles Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic e iSky. Eles utilizam diversas técnicas, incluindo @@@bent pipe e comutação de satélites. Se você deseja uma visão geral de diferentes sistemas e técnicas de satélites, esse artigo é um bom ponto de partida.

Hu e Li, "Satellite-Based Internet: A Tutorial"
O acesso à Internet via satélite é diferente do acesso por linhas terrestres. Não só existe a questão do retardo, mas o roteamento e a comutação também são diferentes. Nesse artigo, os autores examinam algumas questões relacionadas ao uso de satélites para acesso à Internet.

Joel, "Telecommunications and the IEEE Communications Society"
Para conhecer uma história das telecomunicações compacta, mas surpreendentemente completa, começando com o telégrafo e terminando com o padrão 802.11, você deve ler esse artigo. Ele também abrange rádio, telefones, comutação analógica e digital, cabos submarinos, transmissão digital, ATM,

difusão de televisão, satélites, TV a cabo, comunicações ópticas, telefones celulares, comutação de pacotes, a ARPANET e a Internet.

Metcalfe, "Computer/Network Interface Design: Lessons from Arpanet & Ethernet"

Apesar de os engenheiros terem desenvolvido interfaces de rede durante décadas, com freqüência as pessoas se perguntam se eles aprenderam alguma coisa sobre o assunto depois de toda essa experiência. Nesse trabalho, o projetista da Ethernet mostra como criar uma interface de rede e o que fazer com ela após desenvolvê-la. O autor não esconde nada e mostra o que fez de certo e de errado.

Palais, *Fiber Optic Communication*, 3ª edição

Em geral, os livros que tratam da tecnologia de fibra óptica se destinam ao especialista, mas esse livro é mais acessível que a maioria. Ele trata de guias de onda, fontes de luz, detectores de luz, acopladores, modulação, ruído e muitos outros assuntos.

Pandya, "Emerging Mobile and Personal Communications Systems"

Se você deseja uma breve e clara introdução aos sistemas de comunicações pessoais portáteis, leia esse artigo. Uma das nove páginas contém uma lista de 70 acrônimos usados nas outras oito páginas.

Sarikaya, "Packet Mode in Wireless Networks: Overview of Transition to Third Generation"

Toda idéia de redes celulares de terceira geração se refere na realidade à transmissão de dados sem fios. Para ter uma visão geral de como as redes de segunda geração tratam os dados e qual será a evolução até a terceira geração, esse artigo é uma boa fonte de informações. Os tópicos abordados incluem GPRS, IS-95B, WCDMA e CDMA2000.

[T3] 9.1.3 A camada de enlace de dados

Carlson, *PPP Design, Implementation and Debugging*, 2ª edição

Se estiver interessado em obter informações detalhadas sobre todos os protocolos que constituem o conjunto PPP, incluindo o CCP (compactação) e o ECP (criptografia), esse livro é uma boa referência. Existe um enfoque especial no ANU PPP-2.3, uma implementação popular do PPP.

Gravano, *Introduction to Error Control Codes*

Os erros estão presentes em quase todas as comunicações digitais, e foram desenvolvidos muitos tipos de códigos para detectá-los e corrigi-los. Esse livro explica alguns dos mais importantes, desde os códigos de Hamming lineares simples até os mais complexos campos de Galois e @@@códigos de convolução. O autor procura fazê-lo com o mínimo possível de álgebra, mas mesmo isso ainda é muito.

Holzmann, *Design and Validation of Computer Protocols*

Os leitores interessados nos aspectos mais formais dos protocolos de enlace de dados (e semelhantes) devem ler esse volume. A especificação, a modelagem, a precisão e o teste desses protocolos são totalmente descritos no livro.

Peterson e Davie, *Computer Networks: A Systems Approach*

O Capítulo 2 contém material sobre muitas questões de enlace de dados, inclusive enquadramento, detecção de erros, protocolos stop-and-wait (parar e esperar), protocolos de janela deslizante e LANs IEEE 802.

Stallings, *Data and Computer Communications*

O Capítulo 7 lida com a camada de enlace de dados e aborda o controle de fluxo, a detecção de erros e os protocolos básicos de enlace de dados, incluindo stop-and-wait e go back n. Os protocolos do tipo HDLC também são abordados.

[T3] 9.1.4 A subcamada de controle de acesso ao meio

Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps"

Para uma introdução objetiva ao sistema Bluetooth, esse é um excelente ponto de partida. São descritos os protocolos de núcleo e perfis, rádio, @@@piconets e enlaces, seguidos por uma introdução aos diversos protocolos.

Bisdikian, "An Overview of the Bluetooth Wireless Technology"

Como o artigo de Bhagwat (anterior), esse também é um bom ponto de partida para se aprender mais sobre o sistema Bluetooth. As @@@piconets, a pilha de protocolos e os perfis são descritos, como também outros tópicos.

Crow *et al.*, "IEEE 802.11 Wireless Local Area Networks"

Para uma introdução simples à tecnologia e aos protocolos do 802.11, leia esse artigo. É dada ênfase à subcamada MAC. São abordados tanto o controle distribuído quanto o controle centralizado. O artigo conclui com alguns estudos de simulação do desempenho do 802.11 sob diversas condições.

Eklund *et al.*, "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access"

O loop local sem fio padronizado pelo IEEE em 2002 como 802.16 pode revolucionar o serviço telefônico, trazendo a banda larga para o ambiente doméstico. Nessa avaliação, os autores explicam as principais questões tecnológicas relacionadas a esse padrão.

Kapp, "802.11: Leaving the Wire Behind"

Essa breve introdução ao 802.11 aborda os conceitos básicos, os protocolos e os padrões relevantes.

Kleinrock, "On Some Principles of Nomadic Computing and Multi-Access Communications"

O acessos em fio por um canal compartilhado é mais complexo que fazer estações fisicamente conectadas compartilharem um canal. Entre outras questões estão as topologias dinâmicas, o roteamento e o gerenciamento da energia. Essas e outras questões relacionadas ao acesso de canais por dispositivos móveis sem

fios são focalizadas nesse artigo.

Miller e Cummins, *LAN Technologies Explained*

Você precisa saber mais sobre as tecnologias que podem ser usadas em uma LAN? Esse livro aborda a maioria delas, inclusive FDDI e token ring, bem como a sempre popular Ethernet. Embora novas instalações das duas primeiras agora sejam raras, muitas redes atuais ainda as utilizam, e as redes em anel ainda são comuns (por exemplo, SONET).

Perlman, *Interconnections*, 2ª edição

Para quem deseja um texto abrangente, mas divertido, a respeito de pontes, roteadores e roteamento em geral, o livro de Perlman é uma excelente sugestão. O autor elaborou os algoritmos usados na ponte de @@@árvore de amplitude do IEEE 802 e é sem dúvida um dos maiores especialistas mundiais em diversos aspectos de redes.

Webb, "Broadband Fixed Wireless Access"

Esse artigo examina tanto o "porquê" quanto o "como" das redes fixas de banda larga sem fios. A seção do "porquê" afirma que as pessoas não desejam um endereço de correio eletrônico em casa, um endereço de correio eletrônico no trabalho, números de telefones diferentes para casa, trabalho e celular, uma conta de mensagens instantâneas e talvez um ou dois números de fax. Elas desejam um único sistema integrado que funcione em todos os lugares. A ênfase na seção de tecnologia está na camada física, incluindo tópicos como TDD e FDD, modulação adaptável e fixa, e número de portadoras.

[T3] 9.1.5 A camada de rede

Bhatti e Crowcroft, "QoS Sensitive Flows: Issues in IP Packet Handling"

Um dos caminhos para se obter melhor qualidade de serviço em uma rede é programar cuidadosamente as saídas de pacotes de cada roteador. Nesse artigo,

são descritos com algum detalhe diversos algoritmos de programação de execução de pacotes, bem como questões inter-relacionadas.

Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks"

O roteamento em redes *ad hoc* de notebooks que estão próximos um do outro já é bastante difícil sem que tenhamos de nos preocupar também com a qualidade do serviço. Apesar disso, as pessoas se importam com a qualidade do serviço, e portanto é necessário dar alguma atenção a esse tópico. A natureza das redes *ad hoc* e algumas das questões relacionadas ao roteamento e à qualidade do serviço são descritas nesse artigo.

Comer, *Internetworking with TCP/IP*, Vol. 1, 4ª edição

Comer escreveu o trabalho definitivo sobre o conjunto de protocolos TCP/IP. Os Capítulos 4 a 11 tratam do IP e de outros protocolos inter-relacionados que fazem parte da camada de rede. Os outros capítulos lidam principalmente com as camadas mais altas do modelo TCP/IP e também são muito interessantes.

Huitema, *Routing in the Internet*

Se quiser conhecer tudo o que existe sobre o roteamento na Internet, esse é o livro que você está procurando. Todos os algoritmos pronunciáveis (por exemplo, RIP, CIDR e MBONE) e não pronunciáveis (por exemplo, OSPF, IGRP, EGP e BGP) são tratados com riqueza de detalhes. Novas características, como multidifusão (multicast), IP móvel e reserva de recursos, também são estudadas.

Malhotra, *IP Routing*

Se desejar um guia detalhado do roteamento IP, você encontrará nesse livro um bom material sobre o assunto. Os protocolos abordados incluem RIP, RIP-2, IGRP, EIGRP, OSPF e BGP-4.

Metz, "Differentiated Services"

As garantis de qualidade de serviço são importantes para muitas aplicações de multimídia. Serviços integrados e serviços diferenciados são duas abordagens

possíveis para consegui-las. Ambas são discutidas nesse artigo, com ênfase em serviços diferenciados.

Metz, "IP Routers: New Tool for Gigabit Networking"

A maioria das outras referências ao Capítulo 5 tratam de algoritmos de roteamento. Esse é diferente: ele mostra como os roteadores realmente funcionam. Eles passaram por um processo evolucionário, deixando de ser estações de trabalho de uso geral para se transformarem em máquinas de roteamento de uso altamente específico. Se você quiser saber mais sobre o assunto, esse artigo será um bom ponto de partida.

Nemeth *et al.*, *UNIX System Administration Handbook*

Em uma mudança de ritmo, o Capítulo 13 desse livro trata de redes com uma abordagem mais prática que a maior parte de nossas outras referências. Em vez de lidar apenas com os conceitos abstratos, ele oferece conselhos sobre o que fazer se for preciso administrar uma rede real.

Perkins, "Mobile Networking through Mobile IP"

À medida que os dispositivos de computação móveis se tornam cada vez mais comuns, o IP móvel está se tornando mais importante. Esse tutorial oferece uma boa introdução a esse assunto e a outros tópicos inter-relacionados.

Perlman, *Interconnections: Bridges and Routers*, 2ª edição

Nos Capítulos 12 a 15, Perlman descreve muitas questões relacionadas aos projetos de algoritmos de roteamento de unidifusão (unicast) e multidifusão (multicast), tanto no caso de WANs quanto de redes de LANs, e suas implementações em diversos protocolos. Porém, a melhor parte do livro é o Capítulo 18, em que a autora destila seus anos de experiência sobre protocolos de rede em um capítulo informativo e interessante.

Puzmanova, *Routing and Switching: Time of Convergence?*

No final da década de 1990, alguns fornecedores de equipamentos de redes

começaram a denominar todos os equipamentos como switches, enquanto muitos administradores de grandes redes afirmavam estar mudando de roteadores para switches. Como o título implica, esse livro prevê o futuro de roteadores e switches e pergunta se eles estão realmente convergindo.

Royer e Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks"

O algoritmo de roteamento *ad hoc* AODV que descrevemos no Capítulo 5 não é o único algoritmo desse tipo conhecido. Vários outros, inclusive DSDV, CGSR, WRP, DSR, TORA, ABR, DRP e SRP, são discutidos aqui e comparados uns com os outros. É claro que, se estiver planejando criar um novo protocolo de roteamento *ad hoc*, seu primeiro passo deve ser criar um acrônimo de três ou quatro letras.

Stevens, *TCP/IP Illustrated*, Vol. 1

Os Capítulos 3 a 10 apresentam um tratamento completo do IP e de protocolos inter-relacionados (ARP, RARP e ICMP), ilustrado por exemplos.

Striegel e Manimaran, "A Survey of QoS Multicasting Issues"

A multidifusão e a qualidade de serviço são dois tópicos que adquirem uma importância cada vez maior à medida que serviços como o rádio e a televisão pela Internet começam a decolar. Nesse ensaio de pesquisa, os autores descrevem como os algoritmos de roteamento podem levar em conta essas duas questões.

Yang e Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks"

Os autores elaboraram uma taxonomia para algoritmos de controle de congestionamento. As principais categorias são: loop aberto com controle de origem, loop aberto com controle de destino, loop fechado com feedback explícito e loop fechado com feedback implícito. Essa taxonomia é usada para descrever e classificar 23 algoritmos.

[T3] 9.1.6 A camada de transporte

Comer, *Internetworking with TCP/IP*, Vol. 1, 4ª edição

Como mencionamos antes, Comer escreveu o trabalho definitivo sobre o conjunto de protocolos TCP/IP. O Capítulo 12 trata do UDP, enquanto o Capítulo 13 estuda o TCP.

Hall e Cerf, *Internet Core Protocols: The Definitive Guide*

Se você gosta de obter suas informações diretamente da fonte, esse é livro ideal para conhecer mais sobre o TCP. Afinal, Cerf foi um de seus criadores. O Capítulo 7 é uma boa referência sobre o TCP, mostrando como interpretar as informações fornecidas pela análise de protocolo e pelas ferramentas de gerenciamento de rede. Outros capítulos estudam o UDP, o IGMP, o ICMP e o ARP.

Kurose e Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*

O Capítulo 3 trata da camada de transporte e contém uma quantidade razoável de material sobre o UDP e o TCP. Ele também descreve os protocolos stop-and-wait e go back n que examinamos no Capítulo 3.

Mogul, "IP Network Performance"

Apesar do título, esse artigo trata principalmente do TCP e do desempenho de redes em geral, mais do que trata do IP em particular. Há inúmeras orientações e regras práticas bastante úteis.

Peterson e Davie, *Computer Networks: A Systems Approach*

O Capítulo 5 estuda o UDP, o TCP e alguns protocolos relacionados. O desempenho de rede também é abordado resumidamente.

Stevens, *TCP/IP Illustrated*, Vol. 1

Os Capítulos 17 a 24 fornecem um tratamento completo do TCP, ilustrado por exemplos.

[T3] 9.1.7 A camada de aplicação

Bergholz, "Extending Your Markup: An XML Tutorial"

Uma introdução curta e objetiva à XML para iniciantes.

Cardellini *et al.*, *The State-of-the-Art in Locally Distributed Web-Server Systems"*

À medida que a Web se torna mais popular, alguns Web sites precisam ter grandes @@@grupos de servidores (server farms) para lidar com o tráfego. A parte difícil da construção de um grupo de servidores é distribuir a carga entre as máquinas. Esse tutorial discute o assunto com grande profundidade.

Berners-Lee *et al.*, "The World Wide Web"

Uma perspectiva sobre a Web e seu futuro, apresentada por quem a criou e por alguns de seus colegas da CERN. O artigo se concentra na arquitetura da Web, em URLs, HTTP e HTML, bem como em tendências futuras, e compara a Web a outros sistemas distribuídos de informações.

Choudbury *et al.*, "Copyright Protection for Electronic Publishing on Computer Networks"

Apesar de inúmeros livros e artigos descreverem os algoritmos criptográficos, poucos mostram como eles poderiam ser usados para impedir que os usuários distribuam documentos que eles têm permissão para decodificar. Esse artigo apresenta diversos mecanismos que ajudam a proteger os direitos autorais dos criadores na era eletrônica.

Collins, "Carrier Grade Voice over IP"

Se leu o artigo de Varshney *et al.* e agora deseja conhecer todos os detalhes sobre a tecnologia de voz sobre IP usando H.323, esse trabalho será bastante útil. Embora o livro seja longo e detalhado, sua natureza é de tutorial e ele não requer nenhum conhecimento anterior de engenharia de telefonia.

Davison, "A Web Caching Primer"

À medida que a Web cresce, o uso de caches está se tornando crucial para a

obtenção de bom desempenho. Se desejar uma breve introdução aos caches da Web, esse artigo deverá ajudá-lo.

Krishnamurthy e Rexford, *Web Protocols and Practice*

Seria difícil encontrar um livro mais completo do que esse sobre todos os aspectos da Web. Ele aborda clientes, servidores, proxies e caches, como seria de esperar. Contudo, também há capítulos sobre tráfego e medições na Web, bem como capítulos que tratam da pesquisa atual e do aperfeiçoamento da Web.

Rabinovich e Spatscheck, *Web Caching and Replication*

Leia esse livro se desejar um tratamento amplo de caches da Web e replicação. Proxies, caches, pré-busca, redes de entrega de conteúdo, seleção de servidores e muitos outros assuntos são focalizados com grande detalhe.

Shahabi *et al*. "Yima: A Second-Generation Continuous Media Server"

Os servidores de multimídia são sistemas complexos que têm de administrar a programação de execução da CPU, o posicionamento de arquivos de disco, a sincronização de fluxo e muitos outros assuntos. À medida que o tempo passa, as pessoas aprendem a projetá-los cada vez melhor. Uma avaliação arquitetônica de um sistema recente é apresentada nesse artigo.

Tittel *et al*., *Mastering XHTML*

Dois livros em um único volume, abordando a nova linguagem de marcação padrão da Web. Primeiro, há um texto que descreve a linguagem XHTML, concentrando-se principalmente em suas diferenças em relação à HTML comum. Em seguida, há um guia de referência completo tratando de tags, códigos e caracteres especiais utilizados em XHTML 1.0.

Varshney *et al*., "Voice over IP"

Como funciona a voz sobre IP? Essa tecnologia irá substituir a rede pública de telefonia comutada? Leia e descubra.

[T3] 9.1.8 Segurança de redes

Anderson, "Why Cryptosystems Fail"

De acordo com o Anderson, a segurança em sistemas bancários é fraca, mas não devido a intrusos inteligentes que rompem a DES em seus PCs. Os problemas reais variam desde funcionários desonestos (um funcionário de um banco que troca o endereço de um cliente pelo seu próprio endereço para interceptar o cartão do banco e o número de identificação – PIN – do cliente) até erros de programação (fornecendo a todos os clientes o mesmo código PIN). É especialmente interessante observar é a resposta arrogante dada pelos bancos ao se defrontarem com um problema claro: "Nossos sistemas são perfeitos e, portanto, todos os erros devem ser causados por erros do cliente ou por fraude".

Anderson, *Security Engineering*

Até certo ponto, esse livro é uma versão de 600 páginas de "Why Cryptosystems Fail". Ele é mais técnico que *Secrets and Lies*, embora seja menos técnico que *Network Security* (veja a seguir). Depois de uma introdução às técnicas básicas de segurança, capítulos inteiros são dedicados a várias aplicações, inclusive transações bancárias, comando e controle nuclear, impressão de segurança, biometria, segurança física, guerra eletrônica, segurança de telecomunicações, comércio eletrônico e proteção de direitos autorais. A terceira parte do livro trata de política, gerenciamento e avaliação de sistemas.

Artz, "Digital Steganography"

A esteganografia (escrita cifrada) remonta à Grécia antiga, onde a cera de tabuletas em branco era derretida, de forma que fosse possível aplicar mensagens secretas à madeira subjacente antes da reaplicação da cera. Hoje em dia são usadas técnicas diferentes, mas o objetivo é o mesmo. Nesse trabalho são discutidas diversas técnicas modernas para ocultação de informações em imagens, áudio e outras portadoras.

Brands, *Rethinking Public Key Infrastructures and Digital Certificates*

Mais que uma extensa introdução aos certificados digitais, esse livro também é um eficiente trabalho de advocacia. O autor acredita que sistemas atuais de verificação de identidade em papel são antiquados e ineficientes, e argumenta que os certificados digitais podem ser usados para aplicações como votação eletrônica, administração de direitos digitais e até em substituição ao dinheiro. Ele também adverte que, sem PKI e criptografia, a Internet poderia se tornar uma ferramenta de vigilância em ampla escala.

Kaufman *et al.*, *Network Security*, 2ª edição

Esse livro autorizado e engenhoso é a primeira opção para quem deseja informações mais técnicas sobre algoritmos e protocolos de segurança de rede. Algoritmos e protocolos de chave secreta e de chave pública, hash de mensagens, autenticação, Kerberos, PKI, IPsec, SSL/TLS e segurança de correio eletrônico são todos explicados cuidadosamente e em considerável extensão, com muitos exemplos. O Capítulo 26 sobre o folclore de segurança é uma verdadeira jóia. Em segurança, o diabo está nos detalhes. Qualquer pessoa que planeje criar um sistema de segurança que realmente seja usado aprenderá muito com as orientações reais apresentadas nesse capítulo.

Pohlmann, *Firewall Systems*

Os firewalls são a primeira (e a última) linha de defesa da maioria das redes contra atacantes. Esse livro explica como eles funcionam e qual a sua utilidade, desde o mais simples firewall baseado em software criado para proteger um único PC, até os aparelhos avançados de firewalls situados entre uma rede privada e sua conexão para a Internet.

Schneier, *Applied Cryptography*, 2ª edição

Esse monumental compêndio é o pior pesadelo da NSA: um único livro que descreve todos os algoritmos criptográficos conhecidos. Para piorar (ou melhorar,

dependendo de seu ponto de vista), o livro contém a maior parte dos algoritmos sob a forma de programas executáveis (em C). Além disso, são fornecidas mais de 1600 referências à literatura sobre criptografia. Não é um volume para iniciantes mas, se quiser *realmente* manter seus arquivos secretos, você deve ler esse livro.

Schneier, *Secrets and Lies*

Se leu *Applied Cryptography* desde a primeira à última página, você deve saber tudo que é preciso conhecer sobre algoritmos criptográficos. Se ler em seguida *Secrets and Lies* de ponta a ponta (o que pode ser feito em muito menos tempo), você aprenderá que não basta conhecer os algoritmos criptográficos. A maioria das deficiências de segurança não se deve a algoritmos defeituosos ou mesmo a chaves curtas demais, mas sim a falhas no ambiente de segurança. São apresentados inúmeros exemplos de ameaças, ataques, defesas, contra-ataques e muito mais. Se desejar uma descrição não técnica e fascinante da segurança de computadores em seu sentido mais amplo, leia esse livro.

Skoudis, *Counter Hack*

A melhor maneira de deter um hacker é pensar como um hacker. Esse livro mostra como os hackers vêem uma rede e afirma que a segurança deve ser uma função de todo o projeto da rede, não uma reflexão tardia baseada em uma tecnologia específica. O livro cobre quase todos os ataques comuns, inclusive os tipos de "engenharia social" que tiram proveito de usuários que nem sempre estão familiarizados com as medidas de segurança de computadores.

[T2] Bibliografia em ordem alfabética

ABRAMSON, N.: "Internet Access Using VSATs", *IEEE Commun. Magazine*, vol. 38, pp. 60-68, julho de 2000.

ABRAMSON, N.: "Development of the ALOHANET", *IEEE Trans. on Information*

Theory, vol. IT–31, pp. 119–123, março de 1985.

ADAMS, M. e DULCHINOS, D.: "OpenCable", *IEEE Commun. Magazine*, vol. 39, pp. 98–105, junho de 2001.

ALKHATIB, H.S., BAILEY, C., GERLA, M. e McRAE, J.: "Wireless Data Networks: Reaching the Extra Mile", *Computer*, vol. 30, pp. 59–62, dezembro de 1997.

ANDERSON, R.J.: "Free Speech Online and Office", *Computer*, vol. 25, pp. 28–30, junho de 2002.

ANDERSON, R.J.: *Security Engineering*, Nova York: Wiley, 2001.

ANDERSON, R.J.: "The Eternity Service", *Proc. First Int'l Conf. on Theory and Appl. of Cryptology*, CTU Publishing House, 1996.

ANDERSON, R.J.: "Why Cryptosystems Fail", *Commun. of the ACM*, vol. 37, pp. 32–40, novembro de 1994.

ARTZ, D.: "Digital Steganography", *IEEE Internet Computing*, vol. 5, pp. 75–80, 2001.

AZZAM, A.A. e RANSOM, N.: *Broadband Access Technologies*, Nova York: McGraw–Hill, 1999.

BAKNE, A. e BADRINATH, B.R.: "I–TCP: Indirect TCP for Mobile Hosts", *Proc. 15th Int'l Conf. on Distr. Computer Systems*, IEEE, pp. 136–143, 1995.

BALAKRISHNAN, H., SESHAN, S. e KATZ, R.H.: "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", *Proc. ACM Mobile Computing and Networking Conf.*, ACM, pp. 2–11, 1995.

BALLARDIE, T., FRANCIS, P. e CROWCROFT, J.: "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.*, ACM, pp. 85–95, 1993.

BARAKAT, C., ALTMAN, E. e DABBOUS, W.: "On TCP Performance in a Heterogeneous Network: A Survey", *IEEE Commun. Magazine*, vol. 38, pp. 40–46, janeiro de 2000.

BELLAMY, J.: *Digital Telephony*, 3ª edição, Nova York: Wiley, 2000.

BELLMAN, R.E.: *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

BELSNES, D.: "Flow Control in the Packet Switching Networks", *Communications Networks*, Uxbridge, England: Online, pp. 349-361, 1975.

BENNET, C.H. e BRASSARD, G.: "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing*, pp. 175-179, 1984.

BEREZDIVIN, R., BREINIG, R. e TOPP, R.: "Next-Generation Wireless Communication Concepts and Technologies", *IEEE Commun. Magazine*, vol. 40, pp. 108-116, março de 2002.

BERGHEL, H.L.: "Cyber Privacy in the New Millennium", *Computer*, vol. 34, pp. 132-134, janeiro de 2001.

BERGHOLZ, A.: "Extending Your Markup: An XML Tutorial", *IEEE Internet Computing*, vol. 4, pp. 74-79, julho/agosto de 2000.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F. e SECRET, A.: "The World Wide Web", *Commun. of the ACM*, vol. 37, pp. 76-82, agosto de 1994.

BERTSEKAS, D. e GALLAGER, R.: *Data Networks*, 2ª edição, Englewood Cliffs, NJ: Prentice Hall, 1992.

BHAGWAT, P.: "Bluetooth: Technology for Short-Range Wireless Apps", *IEEE Internet Computing*, vol. 5, pp. 96-103, maio/junho de 2001.

BHARGHAVAN, V., DEMERS, A., SHENKER, S. e ZHANG, L.: "MACAW: A Media Access Protocol for Wireless LANs", *Proc. SIGCOMM '94 Conf.*, ACM, pp. 212-225, 1994.

BHATTI, S.N. e CROWCROFT, J.: "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing*, vol. 4, pp. 48-57, julho/agosto de 2000.

BI, Q., ZYSMAN, G.I. e MENKES, H.: "Wireless Mobile Communications at the Start of the 21st Century", *IEEE Commun. Magazine*, vol. 39, pp. 110-116, janeiro de

2001.

BIHAM, E. e SHAMIR, A.: "Differential Cryptanalysis of the Data Encryption Standard", *Proc. 17th Ann. Int'l Cryptology Conf.*, Berlim: Springer-Verlag LNCS 1294, pp. 513-525, 1997.

BIRD, R., GOPAL, I., HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R e YUNG, M.: "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.*, vol. 11, pp. 679-693, junho de 1993.

BIRRELL, A.D. e NELSON, BJ.: "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems*, vol. 2, pp. 39-59, fevereiro de 1984.

BIRYUKOV, A., SHAMIR, A. e WAGNER, D.: "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption*, Berlim: Springer-Verlag LNCS 1978, p. 1, 2000.

BISDIKIAN, C.: "An Overview of the Bluetooth Wireless Technology", *IEEE Commun. Magazine*, vol. 39, pp. 86-94, dezembro de 2001.

BLAZE, M.: "Protocol Failure in the Escrowed Encryption Standard", *Proc. Second ACM Conf. on Computer and Commun. Security*, ACM, pp. 59-67, 1994.

BLAZE, M. e BELLOVIN, S.: "Tapping on My Network Door", *Commun. of the ACM*, vol. 43, p. 136 , outubro de 2000.

BOGINENI, K., SIVALINGAM, K.M. e DOWD, P.W.: "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks", *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 590-604, maio de 1993.

BOLCSKEI, H., PAULRAJ, A.J., HARI, K.V.S. e NABAR, R.U.: "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Commun. Magazine*, vol. 39, pp. 100-108, janeiro de 2001.

BORISOV, N., GOLDBERG, I. e WAGNER, D.: "Intercepting Mobile Communications: The Insecurity of 802.11" *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, pp. 180-188, 2001.

BRANDS, S.: Rethinking Public Key Infrastructures and Digital Certificates, Cambridge, MA: M.I.T. Press, 2000.

BRAY, J. e STURMAN, C.F.: *Bluetooth 1.1: Connect without Cables*, 2ª edição, Upper Saddle River, NJ: Prentice Hall, 2002.

BREYER, R. e RILEY, S.: *Switched, Fast, and Gigabit Ethernet*, Indianapolis, IN: New Riders, 1999.

BROWN, S.: *Implementing Virtual Private Networks*, Nova York: McGraw-Hill, 1999.

BROWN, L., KWAN, M., PIEPRZYK, J. e SEBERRY, J.: "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI", *ASIACRYPT '91 Abstracts*, pp. 25-30, 1991.

BURNETT, S. e PAINE, S.: *RSA Security's Official Guide to Cryptography*, Berkeley, CA: Osborne/McGraw-Hill, 2001.

CAPETANAKIS, J.I.: "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory*, vol. IT-25, pp. 505-515, setembro de 1979.

CARDELLINI, V., CASALICCHIO, E., COLAJANNI, M. e YU, P.S.: "*The State-of the-Art in Locally Distributed Web-Server Systems", ACM Computing Surveys, vol. 34, pp. 263-311, junho de 2002.*

CARLSON, J.: *PPP Design, Implementation and Debugging*, 2ª edição, Boston: Addison-Wesley, 2001.

CERF, V. e KAHN, R.: "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.*, vol. COM-22, pp. 637-648, maio de 1974.

CHAKRABARTI, S.: "QoS Issues in Ad Hoc Wireless Networks", *IEEE Commun. Magazine*, vol. 39, pp. 142-148, fevereiro de 2001.

CHASE, J.S., GALLATIN, A.J. e YOCUM, K.G.: "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine*, vol. 39, pp. 68-75, abril de 2001.

CHEN, B., JAMIESON, K., BALAKRISHNAN, H. e MORRIS, R.: "Span: An Energy-

Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *ACM Wireless Networks*, vol. 8, setembro de 2002.

CHEN, K.-C.: "Medium Access Control of Wireless LANs for Mobile Computing", *IEEE Network Magazine*, vol. 8, pp. 50-63, setembro/outubro de 1994.

CHOUDBURY, A.K., MAXEMCHUK, N.F., PAUL, S. e SCHULZRINNE, H.G.: "Copyright Protection for Electronic Publishing on Computer Networks", *IEEE Network Magazine*, vol. 9, pp. 12-20, maio/junho de 1995.

CHU, Y., RAO, S.G. e ZHANG, H.: "A Case for End System Multicast", *Proc. Int'l Conf. on Measurements and Modeling of Computer Syst.*, ACM, pp. 1-12. 2000.

CLARK, D.D.: "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.*, ACM, pp. 106-114, 1988.

CLARK, D.D.: "Window and Acknowledgement Strategy in TCP", RFC 813, julho de 1982.

CLARK, D.D., DAVIE, B.S., FARBER, D.J., GOPAL, I.S., KADABA, B.K., SINCOSKIE, W.D., SMITH, J.M. e TENNENHOUSE, D.L.: "The Aurora Gigabit Testbed", *Computer Networks and ISDN Systems*, vol. 25, pp. 599-621, janeiro de 1993.

CLARK, D.D., JACOBSON, V., ROMKEY, J. e SALWEN, H.: "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine*, vol. 27, pp. 23-29, junho de 1989.

CLARK, D.D., LAMBERT, M. e ZHANG, L.: "NETBLT: A High Throughput Transport Protocol", *Proc. SIGCOMM '87 Conf.*, ACM, pp. 353-359, 1987.

CLARKE, A.C.: "Extra-Terrestrial Relays", *Wireless World*, 1945.

CLARKE, I., MILLER, S.G., HONG, T.W., SANDBERG, O. e WILEY, B.: "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, vol. 6, pp. 40-49, janeiro/fevereiro de 2002.

COLLINS, D.: *Carrier Grade Voice over IP*, Nova York: McGraw-Hill, 2001.

COLLINS, D. e SMITH, C.: *3G Wireless Networks*, Nova York: McGraw-Hill, 2001.

COMER, D.E.: *The Internet Book*, Englewood Cliffs, NJ: Prentice Hall, 1995.

COMER, D.E.: *Internetworking with TCP/IP*, vol. 1, 4ª edição, Englewood Cliffs, NJ: Prentice Hall, 2000.

COSTA, L.H.M.K., FDIDA, S. e DUARTE, O.C.M.B.: "Hop by Hop Multicast Routing Protocol", *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun.*, ACM, pp. 249-259, 2001.

CRAVER, S.A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D. e FELTEN, E.W.: "Reading Between the Lines: Lessons from the SDMI Challenge", *Proc. 10th USENIX Security Symp.*, USENIX, 2001.

CRESPO, P.M., HONIG, M.L. e SALEHI, J.A.: "Spread-Time Code-Division Multiple Access", *IEEE Trans. on Commun.*, vol. 43, pp. 2139-2148, junho de 1995.

CROW, B.P., WIDJAJA, I, KIM, J.G. e SAKAI, P.T.: "IEEE 802.11 Wireless Local Area Networks", *IEEE Commun. Magazine*, vol. 35, pp. 116-126, setembro de 1997.

CROWCROFT, J., WANG, Z., SMITH, A. e ADAMS, J.: "A Rough Comparison of the IETF and ATM Service Models", *IEEE Network Magazine*, vol. 9, pp. 12-16, novembro/dezembro de 1995.

DABEK, F., BRUNSKILL, E., KAASHOEK, M.F., KARGER, D., MORRIS, R., STOICA, R. e BALAKRISHNAN, H.: "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service", *Proc. 8th Workshop on Hot Topics in Operating Systems*, IEEE, pp. 71-76, 2001a.

DABEK, F., KAASHOEK, M.F., KARGER, D., MORRIS, R. e STOICA, I.: "Wide-Area Cooperative Storage with CFS", *Proc. 18th Symp. on Operating Systems Prin.*, ACM, pp. 202-15, 2001b.

DAEMEN, J. e RIJMEN, V.: *The Design of Rijndael*, Berlim: Springer-Verlag, 2002.

DANTHINE, A.A.S.: "Protocol Representation with Finite-State Models", *IEEE Trans. on Commun.*, vol. COM-28, pp. 632-643, abril de 1980.

DAVIDSON, J. e PETERS, J.: *Voice over IP Fundamentals*, Indianapolis, IN: Cisco

Press, 2000.

DAVIE, B. e REKHTER, Y.: *MPLS Technology and Applications*, San Francisco: Morgan Kaufmann, 2000.

DAVIS, P.T. e McGUFFIN, C.R.: *Wireless Local Area Networks*, Nova York: McGraw-Hill, 1995.

DAVISON, B.D.: "A Web Caching Primer", *IEEE Internet Computing*, vol. 5, pp. 38-45, julho/agosto de 2001.

DAY, J.D.: "The (Un)Revised OSI Reference Model", *Computer Commun. Rev.*, vol. 25, pp. 39-55, outubro de 1995.

DAY, J.D. e ZIMMERMANN, H.: "The OSI Reference Model", *Proc. of the IEEE*, vol. 71, pp. 1334-1340, dezembro de 1983.

DE VRIENDT, J., LAINE, P., LEROUGE, C e XU, X.: "Mobile Network Evolution: A Revolution on the Move", *IEEE Commun. Magazine*, vol. 40, pp. 104-111, abril de 2002.

DEERING, S.E.: "SIP: Simple Internet Protocol", *IEEE Network Magazine*, vol. 7, pp. 16-28, maio/junho de 1993.

DEMERS, A., KESHAV, S. e SHENKER, S.: "Analysis and Simulation of a Fair Queueing Algorithm", *Internetwork: Research and Experience*, vol. 1, pp. 3-26, setembro de 1990.

DENNING, D.E. e SACCO, G.M.: "Timestamps in Key Distribution Protocols", *Commun. of the ACM*, vol. 24, pp. 533-536, agosto de 1981.

DIFFIE, W. e HELLMAN, M.E.: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, pp. 74-84, junho de 1977.

DIFFIE, W. e HELLMAN, M.E.: "New Directions in Cryptography", *IEEE Trans. on Information Theory*, vol. IT-22, pp. 644-654, novembro de 1976.

DIJKSTRA, E.W.: "A Note on Two Problems in Connexion with Graphs", *Numer. Math.*, vol. 1, pp. 269-271, outubro de 1959.

DOBROWSKI, G. e GRISE, D.: *ATM and SONET Basics*, Fuquay-Varina, NC: APDG Telecom Books, 2001.

DONALDSON, G. e JONES, D.: "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine*, vol. 39, pp. 122-126, junho de 2001.

DORFMAN, R.: "Detection of Defective Members of a Large Population", *Annals Math. Statistics*, vol. 14, pp. 436-440, 1943.

DOUFEXI, A., ARMOUR, S., BUTLER, M., NIX, A., BULL, D., McGEEHAN, J. e KARLSSON, P.: "A Comparison of the HIPERLAN/2 and IEEE 802.11A Wireless LAN Standards", *IEEE Commun. Magazine*, vol. 40, pp. 172-180, maio de 2002.

DURAND, A.: "Deploying IPv6", *IEEE Internet Computing*, vol. 5, pp. 79-81, janeiro/fevereiro de 2001.

DUTCHER, B.: *The NAT Handbook*, Nova York: Wiley, 2001.

DUTTA-ROY, A.: "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine*, vol. 39, pp. 81-88, junho de 2001.

EASTTOM, C.: *Learn JavaScript*, Ashburton, U.K.: Wordware Publishing, 2001.

EL GAMAL, T.: "Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory*, vol. IT-31, pp. 469-472, julho de 1985.

ELHANANY, I., KAHANE, M. e SADOT, D.: "Packet Scheduling in Next-Generation Multiterabit Networks", *Computer*, vol. 34, pp. 104-106, abril de 2001.

ELMIRGHANI, J.M.H. e MOUFTAH, H.T.: "Technologies and Architectures for Scalable Dynamic Dense WDM Networks", *IEEE Commun. Magazine*, vol. 38, pp. 58-66, fevereiro de 2000.

FARSEROTU, J. e PRASAD, R.: "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends", *IEEE Commun. Magazine*, vol. 38, pp. 128-133, junho de 2000.

FIORINI, D., CHIANI, M., TRALLI, V. e SALATI., C.: "Can we Trust HDLC?", *Computer*

Commun. Rev., vol. 24, pp. 61-80, outubro de 1994.

FLOYD, S. e JACOBSON, V.: "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking*, vol. 1, pp. 397-413, agosto de 1993.

FLUHRER, S., MANTIN, I. e SHAMIR, A.: "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography*, 2001.

FORD, L.R., Jr. e FULKERSON, D.R.: *Flows in Networks*, Princeton, NJ: Princeton University Press, 1962.

FORD, W. e BAUM, M.S.: *Secure Electronic Commerce*, Upper Saddle River, NJ: Prentice Hall, 2000.

FORMAN, G.H. e ZAHORJAN, J.: "The Challenges of Mobile Computing", *Computer*, vol. 27, pp. 38-47, abril de 1994.

FRANCIS, P.: "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine*, vol. 7, pp. 30-37, maio/junho de 1993.

FRASER, A.G.: "Towards a Universal Data Transport System", in *Advances in Local Area Networks*, Kummerle, K., Tobagi, F. e Limb, J.O. (editores), Nova York: IEEE Press, 1987.

FRENGLE, N.: *I-Mode: A Primer*, Nova York: Hungry Minds, 2002.

GADECKI, C. e HECKERT, C.: *A TM for Dummies*, Nova York: Hungry Minds, 1997.

GARBER, L.: "Will 3G Really Be the Next Big Wireless Technology?", *Computer*, vol. 35, pp. 26-32, janeiro de 2002.

GARFINKEL, S., com SPAFFORD, G.: *Web Security, Privacy, and Commerce*, Sebastopol, CA: O'Reilly, 2002.

GEIER, J.: *Wireless LANs*, 2ª edição, Indianapolis, IN: Sams, 2002.

GEVROS, P., CROWCROFT, J., KIRSTEIN, P. e BHATTI, S.: "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network Magazine*, vol. 15, pp. 16-25, maio/junho de 2001.

GHANI, N. e DIXIT, S.: "TCP/IP Enhancements for Satellite Networks", *IEEE*

Commun. Magazine, vol. 37, pp. 64-72, 1999.

GINSBURG, D.: *ATM: Solutions for Enterprise Networking*, Boston: Addison-Wesley, 1996.

GOODMAN, D.J.: "The Wireless Internet: Promises and Challenges", *Computer*, vol. 33, pp. 36-41, julho de 2000.

GORALSKI, W.J.: *Optical Networking and WDM*, Nova York: McGraw-Hill, 2001.

GORALSKI, W.J.: *SONET*, 2ª edição, Nova York: McGraw-Hill, 2000.

GORALSKI, W.J.: *Introduction to ATM Networking*, Nova York: McGraw-Hill, 1995.

GOSSAIN, H., DE MORAIS CORDEIRO e AGRAWAL, D.P.: "Multicast: Wired to Wireless", *IEEE Commun. Mag.*, vol. 40, pp. 116-123, junho de 2002.

GRAVANO, S.: *Introduction to Error Control Codes*, Oxford, U.K.: Oxford University Press, 2001.

GUO, Y. e CHASKAR, H.: "Class-Based Quality of Service over Air Interfaces in 4G Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pp. 132-137, março de 2002.

HAARTSEN, J.: "The Bluetooth Radio System", *IEEE Personal Commun. Magazine*, vol. 7, pp. 28-36, fevereiro de 2000.

HAC, A.: "Wireless and Cellular Architecture and Services", *IEEE Commun. Magazine*, vol. 33, pp. 98-104, novembro de 1995.

HAC, A. e GUO, L.: "A Scalable Mobile Host Protocol for the Internet", *Int'l J. of Network Mgmt*, vol. 10, pp. 115-134, maio/junho de 2000.

HALL, E. e CERF, V.: *Internet Core Protocols: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

HAMMING, R.W.: "Error Detecting and Error Correcting Codes", *Bell System Tech. J.*, vol. 29, pp. 147-160, abril de 1950.

HANEGAN, K.: *Custom CGI Scripting with Perl*, Nova York: Wiley, 2001.

HARRIS, A.: *JavaScript Programming for the Absolute Beginner*, Premier Press,

2001.

HARTE, L., KELLOGG, S., DREHER, R. e SCHAFFNIT, T.: *The Comprehensive Guide to Wireless Technology*, Fuquay-Varina, NC: APDG Publishing, 2000.

HARTE, L., LEVINE, R. e KIKTA, R.: *3G Wireless Demystified*, Nova York: McGraw-Hill, 2002.

HAWLEY, G.T.: "Historical Perspectives on the U.S. Telephone System", *IEEE Commun. Magazine*, vol. 29, pp. 24-28, março de 1991.

HECHT, J.: "Understanding Fiber Optics", Upper Saddle River, NJ: Prentice Hall, 2001.

HEEGARD, C., COFFEY, J.T., GUMMADI, S., MURPHY, P.A., PROVENCIO, R., ROSSIN, E.J., SCHRUM, S. e SHOEMAKER, M.B.: "High-Performance Wireless Ethernet", *IEEE Commun. Magazine*, vol. 39, pp. 64-73, novembro de 2001.

HELD, G.: *The Complete Modem Reference*, 2ª edição, Nova York: Wiley, 1994.

HELLMAN, M.E.: "A Cryptanalytic Time-Memory Tradeoff", *IEEE Trans. on Information Theory*, vol. IT-26, pp. 401-406, julho de 1980.

HILLS, A.: "Large-Scale Wireless LAN Design", *IEEE Commun. Magazine*, vol. 39, pp. 98-104, novembro de 2001.

HOLZMANN, G.J.: *Design and Validation of Computer Protocols*, Englewood Cliffs, NJ: Prentice Hall, 1991.

HU, Y. e LI, V.O.K.: "Satellite-Based Internet Access", *IEEE Commun. Magazine*, vol. 39, pp. 155-162, março de 2001.

HU, Y.-C. e JOHNSON, D.B.: "Implicit Source Routes for On-Demand Ad Hoc Network Routing", *Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking & Computing*, ACM, pp. 1-10, 2001.

HUANG, V. e ZHUANG, W.: "QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications", *IEEE Commun. Magazine*, vol. 40, pp. 118-125, março de 2002.

HUBER, J.F., WEILER, D. e BRAND, H.: "UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization", *IEEE Commun. Magazine*, vol. 38, pp. 129-136, setembro de 2000.

HUI, J.: "A Broadband Packet Switch for Multi-rate Services", *Proc. Int'l Conf. on Commun.*, IEEE, pp. 782-788, 1987.

HUITEMA, C.: *Routing in the Internet*, Englewood Cliffs, NJ: Prentice Hall, 1995.

HULL, S.: *Content Delivery Networks*, Berkeley, CA: Osborne/McGraw-Hill, 2002.

HUMBLET, P.A., RAMASWAMI, R. e SIVARAJAN, K.N.: "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks", *Proc. SIGCOMM '92 Conf.*, ACM, pp. 2-13, 1992.

HUNTER, D.K. e ANDONOVIC, I.: "Approaches to Optical Internet Packet Switching", *IEEE Commun. Magazine*, vol. 38, pp. 116-122, setembro de 2000.

HUSTON, G.: "TCP in a Wireless World", *IEEE Internet Computing*, vol. 5, pp. 82-84, março/abril de 2001.

IBE, O.C.: *Essentials of ATM Networks and Services*, Boston: Addison-Wesley, 1997.

IRMER, T.: "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine*, vol. 32, pp. 20-28, janeiro de 1994.

IZZO, P.: *Gigabit Networks*, Nova York: Wiley, 2000.

JACOBSON, V.: "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.*, ACM, pp. 314-329, 1988.

JAIN, R.: "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey", *Computer Networks and ISDN Systems*, vol. 27, novembro de 1995.

JAIN, R.: *FDDI Handbook—High-Speed Networking Using Fiber and Other Media*, Boston: Addison-Wesley, 1994.

JAIN, R.: "Congestion Control in Computer Networks: Issues and Trends", *IEEE*

Network Magazine, vol. 4, pp. 24-30, maio/junho de 1990.

JAKOBSSON, M. e WETZEL, S.: "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 200]*, Berlim: Springer-Verlag LNCS 2020, pp. 176-191, 2001.

JOEL, A.: "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine*, 50th Anniversary Issue, pp. 6-14 and 162-167, maio de 2002.

JOHANSSON, P., KAZANTZIDIS, M., KAPOOR, R. e GERLA, M.: "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network Magazine*, vol. 15, pp. 28-37, setembro/outubro de 2001.

JOHNSON, D.B.: "Scalable Support for Transparent Mobile Host Internetworking", *Wireless Networks*, vol. 1, pp. 311-321, outubro de 1995.

JOHNSON, H.W.: *Fast Ethernet—Dawn of a New Network*, Englewood Cliffs, NJ: Prentice Hall, 1996.

JOHNSON, N.F. e JAJODA, S.: "Exploring Steganography: Seeing the Unseen", *Computer*, vol. 31, pp. 26-34, fevereiro de 1998.

KAHN, D.: "Cryptology Goes Public", *IEEE Commun. Magazine*, vol. 18, pp. 19-28, março de 1980.

KAHN, D.: *The Codebreakers*, 2ª edição, Nova York: Macmillan, 1995.

KAMOUN, F. e KLEINROCK, L.: "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks*, vol. 3, pp. 337-353, novembro de 1979.

KAPP, 5.: "802.11: Leaving the Wire Behind", *IEEE Internet Computing*, vol. 6, pp. 82-85, janeiro/fevereiro de 2002.

KARN, P.: "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, pp. 134-140, 1990.

KARTALOPOULOS, S.: *Introduction to DWDM Technology. Data in a Rainbow*, Nova York, NY: IEEE Communications Society, 1999.

KASERA, S.K., HJALMTYSSON, G., TOWLSEY, D.F. e KUROSE, J.F.: "Scalable Reliable Multicast Using Multiple Multicast Channels", *IEEE/ACM Trans. on Networking*, vol. 8, pp. 294-310, 2000.

KATZ, D. e FORD, P.S.: "TUBA: Replacing IP with CLNP", *IEEE Network Magazine*, vol. 7, pp. 38-47, maio/junho de 1993.

KATZENBEISSER, S. e PETITCOLAS, F.A.P.: *Information Hiding Techniques 1km Steganography and Digital Watermarking*, London, Artech House, 2000.

KAUFMAN, C., PERLMAN, R. e SPECINER, M.: *Network Security*, 2ª edição, Englewood Cliffs, NJ: Prentice Hall, 2002.

KELLERER, W., VOGEL, H.-J. e STEINBERG, K.-E.: "A Communication Gateway for Infrastructure-Independent 4G Wireless Access", *IEEE Commun. Magazine*, vol. 40, pp. 126-131, março de 2002.

KERCKHOFF, A.: "La Cryptographie Militaire", *J. des Sciences Militaires*, vol. 9, pp. 5-38, janeiro de 1883 and pp. 161-191,fevereiro de 1883.

KIM, J.B., SUDA, T. e YOSHIMURA, M.: "International Standardization of B-ISDN", *Computer Networks and ISDN Systems*, vol. 27, pp. 5-27, outubro de 1994.

KIPNIS, J.: "Beating the System: Abuses of the Standards Adoptions Process", *IEEE Commun. Magazine*, vol. 38, pp. 102-105, julho de 2000.

KLEINROCK, L.: "On Some Principles of Nomadic Computing and Multi-Access Communications", *IEEE Commun. Magazine*, vol. 38, pp. 46-50, julho de 2000.

KLEINROCK, L. e TOBAGI, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf.*, pp. 187-201, 1975.

KRISHNAMURTHY, B. e REXFORD, J.: *Web Protocols and Practice*, Boston: Addison-Wesley, 2001.

KUMAR, V., KORPI, M. e SENGODAN, S.: *IP Telephony with H.323*, Nova York: Wiley, 2001.

KUROSE, J.F. e ROSS, K.W.: *Computer Networking: A Top-Down Approach Featuring the Internet*, Boston: Addison-Wesley, 2001.

KWOK, T.: "A Vision for Residential Broadband Service: ATM to the Home", *IEEE Network Magazine*, vol. 9, pp. 14-28, setembro/outubro de 1995.

KYAS, 0. e CRAWFORD, G.: *ATM Networks*, Upper Saddle River, NJ: Prentice Hall, 2002.

LAM, C.K.M. e TAN, B.C.Y.: "The Internet Is Changing the Music Industry", *Commun. of the ACM*, vol. 44, pp. 62-66, agosto de 2001.

LANSFORD, J., STEPHENS, A e NEVO, R.: "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence", *IEEE Network Magazine*, vol. 15, pp. 20-27, setembro/outubro de 2001.

LASH, D.A.: *The Web Wizard's Guide to Perl and CGI*, Boston: Addison-Wesley, 2002.

LAUBACH, M.E., FARBER, D.J. e DUKES, S.D.: *Delivering Internet Connections over Cable*, Nova York: Wiley, 2001.

LEE, J.S. e MILLER, L.E.: *CDMA Systems Engineering Handbook*, London: Artech House, 1998.

LEEPER, D.G.: "A Long-Term View of Short-Range Wireless", *Computer*, vol. 34, pp.39-44, junho de 2001.

LEINER, B.M., COLE, R., POSTEL, J. e MILLS, D.: "The DARPA Internet Protocol Suite", *IEEE Commun. Magazine*, vol. 23, pp. 29-34, março de 1985.

LEVINE, D.A. e AKYILDIZ, I.A.: "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology", *IEEE/ACM Trans. on Networking*, vol. 3, pp. 158-168, abril de 1995.

LEVY, S.: "Crypto Rebels", *Wired*, pp. 54-61, maio/junho de 1993.

LI, J., BLAKE, C., DE COUTO, D.S.J., LEE, H.I. e MORRIS, R.: "Capacity of Ad Hoc Wireless Networks", *Proc. 7th Int'l Conf. on Mobile Computing and Networking*,

ACM, pp. 61-69, 2001.

LIN, F., CHU, P. e LIU, M.: "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies", *Proc. SIGCOMM '87 Conf.*, ACM, pp. 126-135, 1987.

LIN, Y.-D., HSU, N.-B. e HWANG, R.-H.: "QoS Routing Granularity in MPLS Networks ,*IEEE Commun. Magazine*, vol. 40, pp. 58-65, junho de 2002.

LISTANI, M., ERAMO, V. e SABELLA, R.: "Architectural and Technological Issues for Future Optical Internet Networks", *IEEE Commun. Magazine*, vol. 38, pp. 82-92, setembro de 2000.

LIU, C.L. e LAYLAND, J.W.: "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, vol. 20, pp. 46-61, janeiro de 1973.

LIVINGSTON, D.: *Essential XML for Web Professionals*, Upper Saddle River, NJ: Prentice Hall, 2002.

LOSHIN, P.: *IPv6 Clearly Explained*, San Francisco: Morgan Kaufmann, 1999.

LOUIS, P.J.: *Broadband Crash Course*, Nova York: McGraw-Hill, 2002.

LU, W.: *Broadband Wireless Mobile: 3G and Beyond*, Nova York: Wiley, 2002.

MACEDONIA, M.R.: "Distributed File Sharing", *Computer*, vol. 33, pp. 99-101, 2000.

MADRUGA, E.L. e GARCIA-LUNA-ACEVES, J.J.: "Scalable Multicasting: the Core-Assisted Mesh Protocol", *Mobile Networks and Applications*, vol. 6, pp. 151-165, abril de 2001.

MALHOTRA, R.: *IP Routing*, Sebastopol, CA: O'Reilly, 2002.

MATSUI, M.: "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology— Eurocrypt '93 Proceedings*, Berlim: Springer-Verlag LNCS 765, pp. 386-397, 1994.

MAUFER, T.A.: *IP Fundamentals*, Upper Saddle River, NJ: Prentice Hall, 1999.

MAZIERES, D. e KAASHOEK, M.F.: "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security*, ACM, pp. 27-36, 1998.

MAZIERES, D., KAMINSKY, M., KAASHOEK, M.F. e WITCHEL, E.: "Separating Key Management from File System Security", *Proc. 17th Symp. on Operating Systems Prin.*, ACM, pp. 124-139, dezembro de 1999.

McFEDRIES, P: *Using JavaScript*, Indianapolis, IN: Que, 2001.

McKENNEY, P.E. e DOVE, K.F.: Efficient Demultiplexing of Incoming TCP Packets", *Proc. SIGCOMM '92 Conf.*, ACM, pp. 269-279, 1992.

MELTZER, K. e MICHALSKI, B.: *Writing CGI Applications with Perl*, Boston: Addison-Wesley, 2001.

MENEZES, A.J. e VANSTONE, S.A.: "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology*, vol. 6, pp. 209-224, 1993.

MERKLE, R.C.: "Fast Software Encryption Functions", *Advances in Cryptology— CRYPTO '90 Proceedings*, Berlim: Springer-Verlag LNCS 473, pp. 476-501, 1991.

MERKLE, R.C. e HELLMAN, M.: "On the Security of Multiple Encryption", *Commun. of the ACM*, vol. 24, pp. 465-467, julho de 1981.

MERKLE, R.C. e HELLMAN, M.: "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory*, vol. IT-24, pp. 525-530, setembro de 1978.

METCALFE, R.M.: "On Mobile Computing", *Byte*, vol. 20, p. 110, setembro de 1995.

METCALFE, R.M.: "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 173-179, fevereiro de 1993.

METCALFE, R.M. e BOGGS, D.R.: "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, pp. 395-404, julho de 1976.

METZ, C: "Interconnecting ISP Networks", *IEEE Internet Computing*, vol. 5, pp. 74-

80, março/abril de 2001.

METZ, C.: "Differentiated Services", *IEEE Multimedia Magazine*, vol. 7, pp. 84-90, julho/setembro de 2000.

METZ, C.: "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing*, vol. 2, pp. 14-18, novembro/dezembro de 1998.

MILLER, B.A. e BISDIKIAN, C.,: *Bluetooth Revealed*, Upper Saddle River, NJ: Prentice Hall, 2001.

MILLER, P. e CUMMINS, M.: *LAN Technologies Explained*, Woburn, MA: Butterworth-Heinemann, 2000.

MINOLI, D.: *Video Dialtone Technology*, Nova York: McGraw-Hill, 1995.

MINOLI, D. e VITELLA, M.: *ATM & Cell Relay for Corporate Environments*, Nova York: McGraw-Hill, 1994.

MISHRA, P.P. e KANAKIA, H.: "A Hop by Hop Rate-Based Congestion Control Scheme", *Proc. SIGCOMM '92 Conf.*, ACM, pp. 112-123, 1992.

MISRA, A., DAS, S., DUTTA, A., McAULEY, A. e DAS, S.: "IDMP-Based Fast Handoffs and Paging in IP-Based 4G Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pp. 138-145, março de 2002.

MOGUL, J.C.: "IP Network Performance", em *Internet System Handbook*, Lynch, D.C. e Rose, M.T. (editores), Boston: Addison-Wesley, pp. 575-675, 1993.

MOK, A.K. e WARD, S.A.: "Distributed Broadcast Channel Access", *Computer Net-works*, vol. 3, pp. 327-335, novembro de 1979.

MOY, J.: "Multicast Routing Extensions", *Commun. of the ACM*, vol. 37, pp. 61-66, agosto de 1994.

MULLINS, J.: "Making Unbreakable Code", *IEEE Spectrum*, pp. 40-45, maio de 2002.

NAGLE, J.: "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.*, vol. COM-35, pp. 435-438, abril de 1987.

NAGLE, J.: "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.*, vol. 14, pp. 11-17, outubro de 1984.

NARAYANASWAMI, C., KAMIJOH, N., RAGHUNATH, M., INOUE, T., CIPOLLA, T., SANFORD, J., SCHLIG, E., VENTKITESWARAN, S., GUNIGUNTALA, D., KULKARNI, V. e YAMAZAKI, K.: "IBM's Linux Watch: The Challenge of Miniaturization", *Computer*, vol. 35, pp. 33-41, janeiro de 2002.

NAUGHTON, J.: "A Brief History of the Future", Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R.M. e SCHROEDER, M.D.: "Authentication Revisited", *Operating Systems Rev.*, vol. 21, p. 7, janeiro de 1987.

NEEDHAM, R.M. e SCHROEDER, M.D.: "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM*, vol. 21, pp. 993-999, dezembro de 1978.

NELAKUDITI, S. e ZHANG, Z.-L.: "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine* vol. 40, pp. 66-71, junho de 2002.

NEMETH, E., SNYDER, G., SEEBASS, S. e HEIN, T.R.: *UNIX System Administration Handbook*, 3ª edição, Englewood Cliffs, NJ: Prentice Hall, 2000.

NICHOLS, R.K. e LEKKAS, P.C.: *Wireless Security*, Nova York: McGraw-Hill, 2002.

NIST: "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

O'HARA, B. e PETRICK, A.: *802.11 Handbook: A Designer's Companion*, Nova York: IEEE Press, 1999.

OTWAY, D. e REES, 0.: "Efficient and Timely Mutual Authentication", *Operating Systems Rev.*, pp. 8-10, janeiro de 1987.

OVADIA, S.: *Broadband Cable TV Access Networks: from Technologies to Applications*, Upper Saddle River, NJ: Prentice Hall, 2001.

PALAIS, J.C.: *Fiber Optic Commun.*, 3ª edição, Englewood Cliffs, NJ: Prentice Hall,

1992.

PAN, D.: "A Tutorial on MPEG/Audio Compression", *IEEE Multimedia Magazine*, vol. 2, pp. 60-74, verão de 1995.

PANDYA, R.: "Emerging Mobile and Personal Communication Systems", *IEEE Commun. Magazine*, vol. 33, pp. 44-52, junho de 1995.

PARAMESWARAN, M., SUSARLA, A. e WHINSTON, A.B.: "P2P Networking: An Information-Sharing Alternative", *Computer*, vol. 34, pp. 31-38, julho de 2001.

PARK, J.S. e SANDHU, R.: "Secure Cookies on the Web", *IEEE Internet Computing*, vol. 4, pp. 36-44, julho/agosto de 2000.

PARTRIDGE, C., HUGHES, J. e STONE, J.: "Performance of Checksums and CRCs over Real Data", *Proc. S1GCOMM '95 Conf.*, ACM, pp. 68-76, 1995.

PAXSON, V.: "Growth Trends in Wide-Area TCP Connections", *IEEE Network Maga-zine*, vol. 8, pp. 8-17, julho/agosto de 1994.

PAXSON, V. e FLOYD, S.: "Wide-Area Traffic: The Failure of Poisson Modeling", *Proc. SIGCOMM '94 Conf.*, ACM, pp. 257-268, 1995.

PEPELNJAK, I. e GUICHARD, J.: *MPLS and VPN Architectures*, Indianapolis, IN: Cisco Press, 2001.

PERKINS, C.E.: *RTP: Audio and Video for the Internet*, Boston: Addison-Wesley, 2002.

PERKINS, C.E. (editor): *Ad Hoc Networking*, Boston: Addison-Wesley, 2001.

PERKINS, C.E.: *Mobile IP Design Principles and Practices*, Upper Saddle River, NJ: Prentice Hall, 1998a.

PERKINS, C.E.: "Mobile Networking in the Internet", *Mobile Networks and Applications*, vol. 3, pp. 319-334, 1998b.

PERKINS, C.E.: "Mobile Networking through Mobile IP", *IEEE Internet Computing*, vol. 2, pp. 58-69, janeiro/fevereiro de 1998c.

PERKINS, C.E. e ROYER, E.: "The Ad Hoc On-Demand Distance-Vector Protocol",

em *Ad Hoc Networking*, edited by C. Perkins, Boston: Addison-Wesley, 2001.

PERKINS, C.E. e ROYER, E.: "Ad-hoc On-Demand Distance Vector Routing", *Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications*, IEEE, pp. 90-100, 1999.

PERLMAN, R.: *Interconnections*, 2ª edição, Boston: Addison-Wesley, 2000.

PERLMAN, R.: *Network Layer Protocols with Byzantine Robustness*, tese de doutorado, M.I.T., 1988.

PERLMAN, R. e KAUFMAN, C.: "Key Exchange in IPsec", *IEEE Internet Computing*, vol. 4, pp. 50-56, novembro/dezembro de 2000.

PETERSON, L.L. e DAVIE, B.S.: *Computer Networks: A Systems Approach*, San Francisco: Morgan Kaufmann, 2000.

PETERSON, W.W. e BROWN, D.T.: "Cyclic Codes for Error Detection", *Proc. IRE*, vol. 49, pp. 228-235, janeiro de 1961.

PICKHOLTZ, R.L., SCHILLING, D.L. e MILSTEIN. L.B.: "Theory of Spread Spectrum Communication — A Tutorial", *IEEE Trans. on Commun.*, vol. COM-30, pp. 855-884, maio de 1982.

PIERRE, G., KUZ, I., VAN STEEN, M., TANENBAUM, A.S.: "Differentiated Strategies for Replicating Web Documents", *Computer Commun.*, vol. 24, pp. 232-240, fevereiro de 2001.

PIERRE, G., VAN STEEN, M. e TANENBAUM, A.S.: "Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Trans. on Computers*, vol. 51, pp., junho de 2002.

PISCITELLO, D.M. e CHAPIN, A.L.: *Open Systems Networking: TCP/IP and OSI*, Boston: Addison-Wesley, 1993.

PITT, D.A.: "Bridging—The Double Standard", *IEEE Network Magazine*, vol. 2, pp. 94-95, janeiro de 1988.

PIVA, A., BARTOLINI, F. e BARNI, M.: "Managing Copyrights in Open Networks",

IEEE Internet Computing, vol. 6, pp. 18-26, maio/junho de 2002.

POHLMANN, N.: *Firewall Systems*, Bonn, Alemanha: MITP-Verlag, 2001.

PUZMANOVA, R.: *Routing and Switching: Time of Convergence?*, London: Addison Wesley, 2002.

RABINOVICH, M. e SPATSCHECK, 0,: *Web Caching and Replication*, Boston: Addison-Wesley, 2002.

RAJU, J. e GARCIA-LUNA-ACEVES, J.J.: "Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks", *ACM Computer Communications Review*, vol. 31, October 2001.

RAMANATHAN, R. e REDI, J.: "A Brief Overview of Ad Hoc Networks: Challenges and Directions", *IEEE Commun. Magazine*, 50th Anniversary Issue, pp. 20-22, maio de 2002.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R. e SHENKER, S.: "A Scalable Content-Addressable Network", *Proc. SIGCOMM '01 Conf.*, ACM, pp. 1161-1172, 2001.

RIVEST, R.L.: "The MD5 Message-Digest Algorithm", RFC 1320, abril de 1992.

RIVEST, R.L. e SHAMIR, A.: "How to Expose an Eavesdropper", *Commun. of the ACM*, vol. 27, pp. 393-395, abril de 1984.

RIVEST, R.L., SHAMIR, A. e ADLEMAN, L.: "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM*, vol. 21, pp. 120-126, fevereiro de 1978.

ROBERTS, L.G.: "Dynamic Allocation of Satellite Capacity through Packet Reservation", *Proc. NCC*, AFIPS, pp. 711-716, 1973.

ROBERTS, L.G.: "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", *Proc. Spring Joint Computer Conference*, AFIPS, pp. 295-298, 1972.

ROBERTS, L.G.: "Multiple Computer Networks and Intercomputer Communication",

Proc. First Symp. on Operating Systems Prin., ACM, 1967.

ROSE, M.T.: *The Simple Book*, Englewood Cliffs, NJ: Prentice Hall, 1994.

ROSE, M.T.: *The Internet Message*, Englewood Cliffs, NJ: Prentice Hall, 1993.

ROSE, M.T. e McCLOGHRIE, K.: *How to Manage Your Network Using SNMP*, Englewood Cliffs, NJ: Prentice Hall, 1995.

ROWSTRON, A. e DRUSCHEL, P.: "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility", *Proc. 18th Symp. on Operating Systems Prin.*, ACM, pp. 188-201, 2001a.

ROWSTRON, A. e DRUSCHEL, P.: "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms*, ACM/IFIP, 200 lb.

ROYER, E.M. e TOH, C.-K.: "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Commun. Magazine*, vol. 6, pp. 46-55, abril de 1999.

RUIZ-SANCHEZ, M.A., BIERSACK, E.W. e DABBOUS, W.: "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine*, vol. 15, pp. 8-23, março/abril de 2001.

SAIRAM, K.V.S.S.S.S., GUNASEKARAN, N. e REDDY, S.R.: "Bluetooth in Wireless Communication", *IEEE Commun. Mag.*, vol. 40, pp. 90-96, junho de 2002.

SALTZER, J.H., REED, D.P. e CLARK, D.D.: "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems*, vol. 2, pp. 277-288, novembro de 1984.

SANDERSON, D.W. e DOUGHERTY, D.: *Smileys*, Sebastopol, CA: O'Reilly, 1993.

SARI, H., VANHAVERBEKE, F. e MOENECLAEY, M.: "Extending the Capacity of Multiple Access Channels", *IEEE Commun. Magazine*, vol. 38, pp. 74-82, janeiro de 2000.

SARIKAYA, B.: "Packet Mode in Wireless Networks: Overview of Transition to Third

Generation", *IEEE Commun. Magazine*, vol. 38, pp. 164-172, setembro de 2000.

SCHNEIER, B.: *Secrets and Lies*, Nova York: Wiley, 2000.

SCHNEIER, B.: *Applied Cryptography*, 2ª edição, Nova York: Wiley, 1996.

SCHNEIER, B.: *E-Mail Security*, Nova York: Wiley, 1995.

SCHNEIER, B.: "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]", *Proc. of the Cambridge Security Workshop*, Berlim: Springer-Verlag LNCS 809, pp. 191-204, 1994.

SCHNORR, C.P.: "Efficient Signature Generation for Smart Cards", *Journal of Cryptology*, vol. 4, pp. 161-174, 1991.

SCHOLTZ, R.A.: "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.*, vol. COM-30, pp. 822-854, maio de 1982.

SCOTT, R.: "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia*, vol. 9, pp. 75-90, janeiro de 1985.

SEIFERT, R.: *The Switch Book*, Boston: Addison-Wesley, 2000.

SEIFERT, R.: *Gigabit Ethernet*, Boston: Addison-Wesley, 1998.

SENN, J.A.: "The Emergence of M-Commerce", *Computer*, vol. 33, pp. 148-150, dezembro de 2000.

SERJANTOV, A.: "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems*, Berlim: Springer-Verlag LNCS, 2002.

SEVERANCE, C.: "IEEE 802.11: Wireless Is Coming Home", *Computer*, vol. 32, pp. 126-127, novembro de 1999.

SHAHABI, C., ZIMMERMANN, R., FU, K. e YAO, S.-Y.D.: "YIMA: A Second Generation Continuous Media Server", *Computer*, vol. 35, pp. 56-64, junho de 2002.

SHANNON, C.: "A Mathematical Theory of Communication", *Bell System Tech. J.*, vol. 27, pp. 379-423, julho de 1948; and pp. 623-656, outubro de 1948.

SHEPARD, S.: *SONET/SDH Demystified*, Nova York: McGraw-Hill, 2001.

SHREEDHAR, M. e VARGHESE, G.: "Efficient Fair Queueing Using Deficit Round

Robin", *Proc. SIGCOMM '95 Conf.*, ACM, pp. 231-243, 1995.

SKOUDIS, E.: *Counter Hack*, Upper Saddle River, NJ: Prentice Hall, 2002.

SMITH, D.K. e ALEXANDER, R.C.: *Fumbling the Future*, Nova York: William Morrow, 1988.

SMITH, R.W.: *Broadband Internet Connections*, Boston: Addison Wesley, 2002.

SNOEREN, A.C. e BALAKRISHNAN, H.: "An End-to-End Approach to Host Mobility", *Int'l Conf. on Mobile Computing and Networking* ,ACM, pp. 155-166, 2000.

SOBEL, D.L.: "Will Carnivore Devour Online Privacy", *Computer*, vol. 34, pp. 87-88, maio de 2001.

SOLOMON, J.D.: *Mobile IP: The Internet Unplugged*, Upper Saddle River, NJ: Prentice Hall, 1998.

SPOHN, M. e GARCIA-LUNA-ACEVES, J.J.: "Neighborhood Aware Source Routing", *Proc. ACM MobiHoc 2001*, ACM, pp. 2001.

SPURGEON, C.E.: *Ethernet: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W.: *Data and Computer Communications*, 6th ed., Upper Saddle River, NJ: Prentice Hall, 2000.

STEINMETZ, R. e NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 1: Media Coding and Content Processing*, Upper Saddle River, NJ: Prentice Hall, 2002.

STEINMETZ, R. e NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 2: Media Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2003a.

STEINMETZ, R. e NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 3: Documents, Security, and Applications*, Upper Saddle River, NJ: Prentice Hall, 2003b.

STEINER, J.G., NEUMAN, B.C. e SCHILLER, J.I.: "Kerberos: An Authentication Service for Open Network Systems", *Proc. Winter USENIX Conf.*, USENIX, pp. 191-201, 1988.

STEVENS, W.R.: *UNIX Network Programming*, *Volume 1: Networking APIs - Sockets and XTI.* Upper Saddle River. NJ: Prentice Hall, 1997.

STEVENS, W.R.: *TCP/IP Illustrated*, Vol. 1, Boston: Addison-Wesley, 1994.

STEWART, R. e METZ, C.: "SCTP: New Transport Protocol for TCP/IP", *IEEE Internet Computing*, vol. 5, pp. 64-69, novembro/dezembro de 2001.

STINSON, D.R.: *Cryptography Theory and Practice*, 2ª edição, Boca Raton, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M.F. e BALAKRISHNAN, H.: "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM '01 Conf.*, ACM, pp. 149-160, 2001.

STRIEGEL, A. e MANIMARAN, G.: "A Survey of QoS Multicasting Issues", *IEEE Commun. Mag.*, vol. 40, pp. 82-87, junho de 2002.

STUBBLEFIELD, A., IOANNIDIS, J. e RUBIN, A.D.: "Using the Fluhrer, Mantin e Shamir Attack to Break WEP", *Proc Network and Distributed Systems Security Symp.*, ISOC, pp. 1-11, 2002.

SUMMERS, C.K.: *ADSL: Standards, Implementation, and Architecture*, Boca Raton, FL: CRC Press, 1999.

SUNSHINE, C.A. e DALAL, Y.K.: "Connection Management in Transport Protocols", *Computer Networks*, vol. 2, pp. 454-473, 1978.

TANENBAUM, A.S.: *Modern Operating Systems*, Upper Saddle River, NJ: Prentice Hall, 2001.

TANENBAUM, A.S. e VAN STEEN, M.: *Distributed Systems: Principles and Paradigms*, Upper Saddle River, NJ: Prentice Hall, 2002.

TEGER, S. e WAKS, D.J.: "End-User Perspectives on Home Networking", *IEEE Commun. Magazine*, vol. 40, pp. 114-119, abril de 2002.

THYAGARAJAN, A.S. e DEERING, S.E.: "Hierarchical Distance-Vector Multicast Routing for the MBone", *Proc. SIGCOMM '95 Conf.*, ACM, pp. 60-66, 1995.

TITTEL, E., VALENTINE, C., BURMEISTER, M. e DYKES, L.: *Mastering XHTML*, Alameda, CA: Sybex, 2001.

TOKORO, M. e TAMARU, K.: "Acknowledging Ethernet", *Compcon*, IEEE, pp. 320-325, Fall 1977.

TOMLINSON, R.S.: "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, pp. 11-23, 1975.

TSENG, Y.-C., WU, S.-L., LIAO, W.-H. e CHAO, C.-M.: "Location Awareness in Ad Hoc Wireless Mobile Networks", *Computer*, vol. 34, pp. 46-51, 2001.

TUCHMAN, W.: "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum*, vol. 16, pp. 40-41, julho de 1979.

TURNER, J.S.: "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine*, vol. 24, pp. 8-15, outubro de 1986.

VACCA, J.R.: *I-Mode Crash Course*, Nova York: McGraw-Hill, 2002.

VALADE, J.,: *PHP & MySQL for Dummies*, Nova York: Hungry Minds, 2002.

VARGHESE, G. e LAUCK, T.: "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.*, ACM, pp. 25-38, 1987.

VARSHNEY, U., SNOW, A., McGIVERN, M. e HOWARD, C.: "Voice over IP", *Commun. of the ACM*, vol. 45, pp. 89-96, 2002.

VARSHNEY, U. e VETTER, R.: "Emerging Mobile and Wireless Networks", *Commun. of the ACM*, vol. 43, pp. 73-81, junho de 2000.

VETTER, P., GODERIS, D., VERPOOTEN, L. e GRANGER, A.: "Systems Aspects of APON/VDSL Deployment", *IEEE Commun. Magazine*, vol. 38, pp. 66-72, maio de 2000.

WADDINGTON, D.G. e CHANG, F.: "Realizing the Transition to IPv6", *IEEE Commun. Mag.*, vol. 40, pp. 138-148, junho de 2002.

WALDMAN, M., RUBIN, A.D. e CRANOR, L.F.: "Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System", *Proc. Ninth USENIX Security Symp.*, USENIX, pp. 59-72, 2000.

WANG, Y. e CHEN, W.: "Supporting IP Multicast for Mobile Hosts", *Mobile Networks and Applications*, vol. 6, pp. 57-66, janeiro/fevereiro de 2001.

WANG, Z.: *Internet QoS*, San Francisco: Morgan Kaufmann, 2001.

WARNEKE, B., LAST, M., LIEBOWITZ, B. e PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *Computer*, vol. 34, pp. 44-51, janeiro de 2001.

WAYNER, P.: *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking*, 2ª edição, San Francisco: Morgan Kaufmann, 2002.

WEBB, W.: "Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment", *IEEE Commun. Magazine*, vol. 39, pp. 115-121, setembro de 2001.

WEISER, M.: "Whatever Happened to the Next Generation Internet?", *Commun. of the ACM*, vol. 44, pp. 61-68, setembro de 2001.

WELTMAN, R. e DAHBURA, T.: *LDAP Programming with Java*, Boston: Addison-Wesley, 2000.

WESSELS, D.: *Web Caching*, Sebastopol, CA: O'Reilly, 2001.

WETTEROTH, D.: *OSI Reference Model for Telecommunications*, Nova York: McGraw-Hill, 2001.

WILJAKKA, J.: "Transition to Ipv6 in GPRS and WCDMA Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pp. 134-140, abril de 2002.

WILLIAMSON, H.: *XML: The Complete Reference*, Nova York: McGraw-Hill, 2001.

WILLINGER, W., TAQQU, M.S., SHERMAN, R. e WILSON, D.V.: "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proc. SIGCOMM '95 Conf.*, ACM, pp. 100-113, 1995.

WRIGHT, D.J.: *Voice over Packet Networks*, Nova York: Wiley, 2001.

WYLIE, J., BIGRIGG, M.W., STRUNK, J.D., GANGER, G.R., KILICCOTE, H. e KHOSLA, P.K.: "Survivable Information Storage Systems", *Computer*, vol. 33, pp. 61-68,

agosto de 2000.

XYLOMENOS, G., POLYZOS, G.C., MAHONEN, P. e SAARANEN, M.: "TCP Performance Issues over Wireless Links", *IEEE Commun. Magazine*, vol. 39, pp. 52-58, abril de 2001.

YANG, C.-Q. e REDDY, A.V.S.: "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", *IEEE Network Magazine*, vol. 9, pp. 34-45, julho/agosto de 1995.

YUVAL, G.: "How to Swindle Rabin", *Cryptologia*, vol. 3, pp. 187-190, julho de 1979.

ZACKS, M.: "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing*, vol. 5, pp. 8-9, novembro/dezembro de 2001.

ZADEH, A.N., JABBARI, B., PICKHOLTZ, R. e VOJCIC, B.: "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag.*, vol. 40, pp. 149-157, junho de 2002.

ZHANG, L.: "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine*, vol. 2, pp. 44-48, janeiro/fevereiro de 1988.

ZHANG, L.: "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine*, vol. 7, pp. 8-18, setembro/outubro de 1993.

ZHANG, Y. e RYU, B.: "Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications*, vol. 6, pp. 81-94, janeiro/fevereiro de 2001.

ZIMMERMANN, P.R.: *The Official PGP User's Guide*, Cambridge, MA: M.I.T. Press, 1995a.

ZIMMERMANN, P.R.: *PGP: Source Code and Internals*, Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Boston: Addison-Wesley, 1949.

ZIV, J. e LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression",

IEEE Trans. on Information Theory, vol. IT-23, pp. 337-343, maio de 1977.

[TA]Índice

[T2]Números

A

Q